2020/06/20

ODE インストールガイド

ver.2.7

目次

1．Windows 用セットアップ 2

1－1．開発環境が Visual Studio の場合 2

1－2．開発環境が GCC の場合 8

1－3．　Q&A 10

2．Mac 用セットアップ 11

2－1．ターミナルを用いた実行方法（基礎） 11

2－2．　開発環境が Xcode の場合 12

2－3．　開発環境がターミナルの場合 19

2－4．Q&A 19

3．Linux 用セットアップ（参考） 20

3－1．ターミナルを用いた実行方法（基礎） 20

1. Windows 用セットアップ

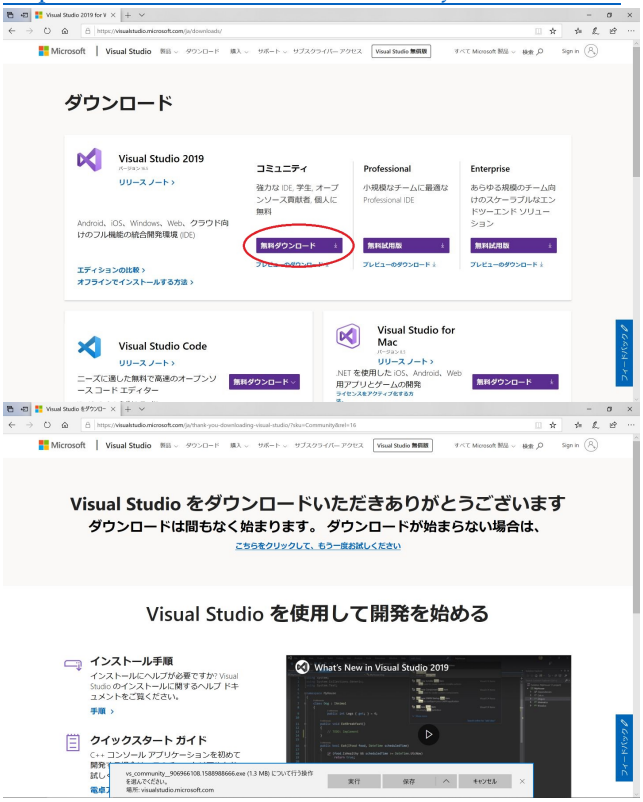
・開発環境：　GCC または Visual Studio

1−1. 開発環境が Visual Studio の場合

（1）　Visual Studio 2019 Community のダウンロード:

以下の URL から，Visual Studio 2019 Community を無料でダウンロードする.

https://visualstudio.microsoft.com/ja/downloads/



旧バージョンを使用したい場合，以下の URL から，Visual Studio Dev Essentials（無償）に，Microsoft アカウントでログインする.

https://visualstudio.microsoft.com/ja/dev-essentials/

以下の URL から，Visual Studio のセットアップファイルをダウンロードする.

最新版の場合

https://visualstudio.microsoft.com/ja/free-developer-offers/→「ダウンロード」

過去版の場合

https://visualstudio.microsoft.com/ja/vs/older-downloads/→ダウンロードしたいバージョンのプルダウンメニューを開く→「ダウンロード」→ダウンロードしたいソフトのシステム，言語，ファイル形式を選択し，「Download」

（2）　Visual Studio のインストール:

ダウンロードしたセットアップファイルを実行し，Visual Studio をインストールする.

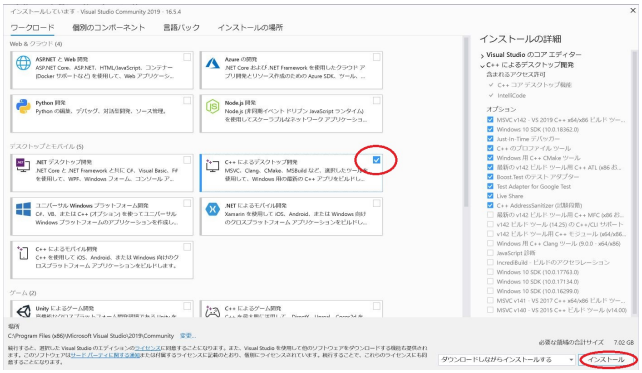
① Visual Studio Installer にて，「Visual Studio Community」を選択する.

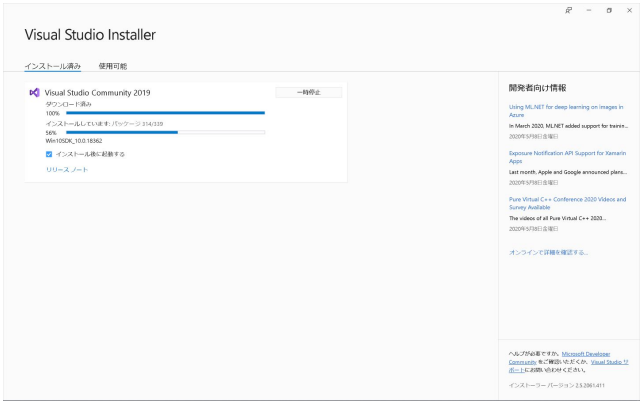
② 「ワークロード」タブより，「ユニバーサル Windows プラットフォーム開発」「C＋＋によるデスクトップ開発」「C＋＋によるモバイル開発」「C＋＋による Linux 開発」を選択しチェックを入れる.
全部入れると 24GB 以上なので要注意. 最低限, 必須の「C＋＋によるデスクトップ開発」だけで 7GB.

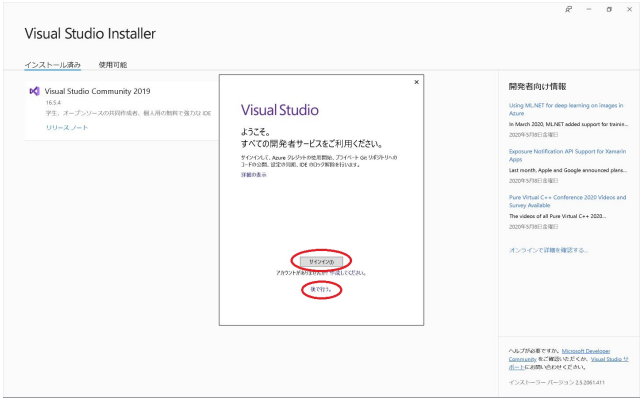
③ 「個別のコンポーネント」はそのままで良い.

④ 任意で，「言語パック」，「インストールの場所」を自分の好きな言語，場所に選択して良い.

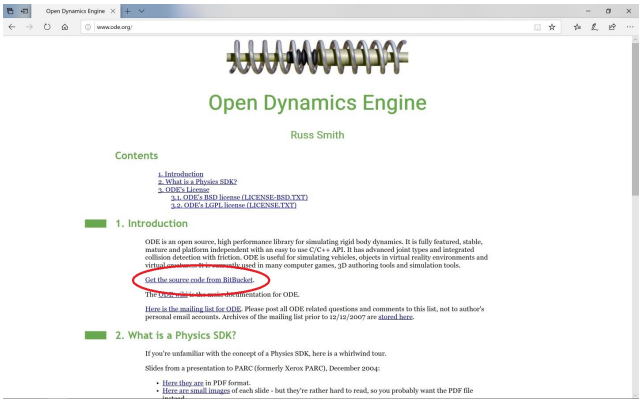
⑤ 下部の，「インストール」を選択し，インストールを行う．
⑥ インストール後は「サインイン」もしくは「後で行う」を選択し，終了．

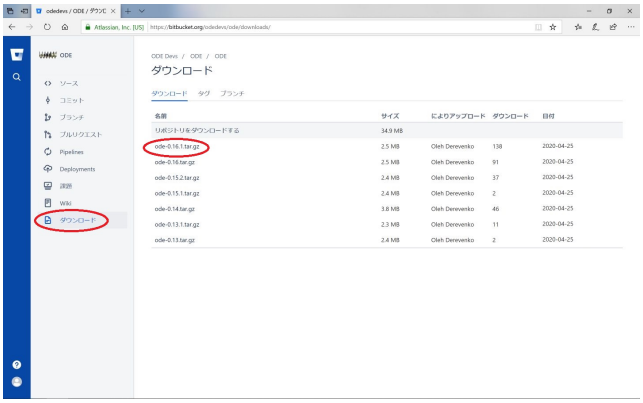






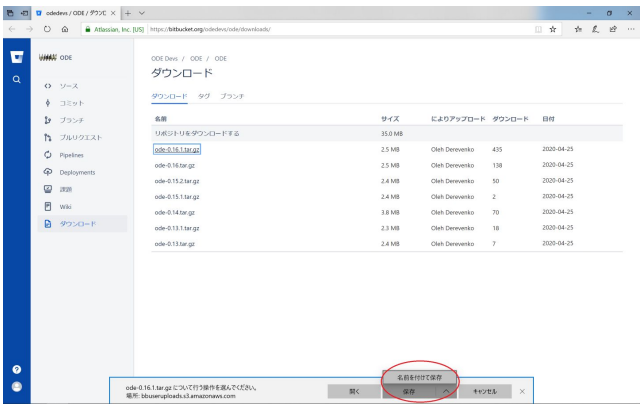
（3） Open Dynamics Engine (ODE)のダウンロード:
ODEソースファイルを以下のURLからode-0.16.1.tar.gzをダウンロードし，Cドライブ直下にodeというフォルダ名で解凍する．
（ア）ブラウザで以下の URL にアクセスし，ode-0.16.1.tar.gz をダウンロード（クリック）
http://www.ode.org/ → https://bitbucket.org/odedevs/ode/downloads/

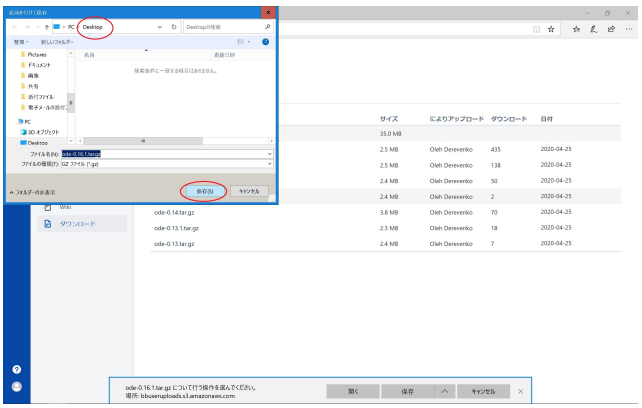




（2014年 Ver.0.13（Russ Smith製） 2020年 Ver.0.16.1（Oleh Derevenko製））

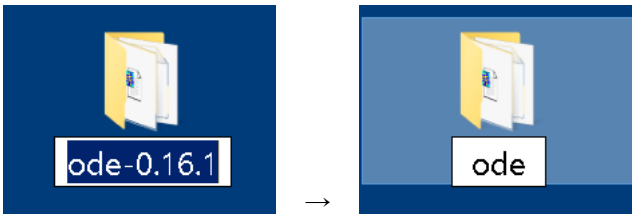
② 保存の中から，「名前を付けて保存」を選択し，「Desktop」フォルダを選択．





③ デスクトップに保存されたそのファイル ode-0.16.1.tar.gz のアイコンをダブルクリック（もしくは右クリックして出てきたメニューから解凍を選択）．解凍できない場合は下記参照．
*.tar.gzファイルは，たとえばフリーソフトの7-Zip（https://sevenzip.osdn.jp/）や Lhaplus（https://forest.watch.impress.co.jp/library/software/lhaplus/）などで解凍可能．

④ 解凍されたフォルダ名「ode-0.16.1」をクリックし，「ode」にする．

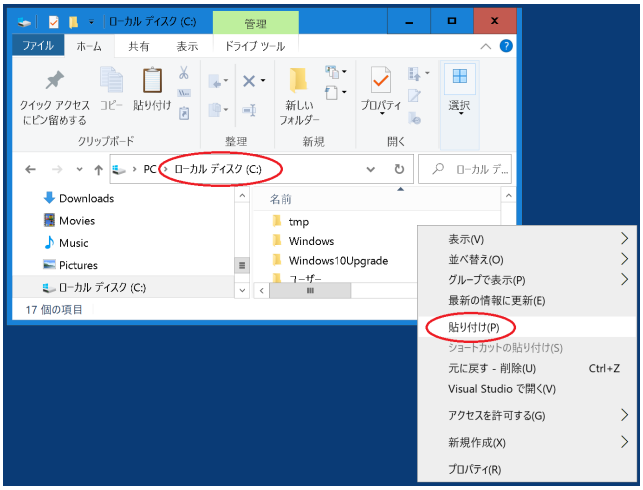


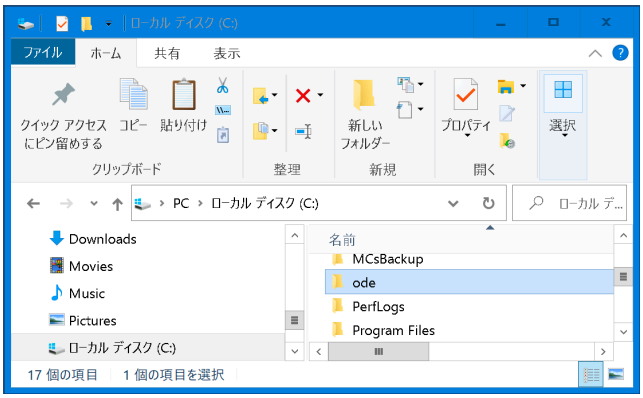
⑤ フォルダを右クリックして出てきたメニューから「切り取り」を選択（フォルダを選択した状態で Ctrl＋x でも可）．

⑥ エクスプローラ（黄色いアイコン 

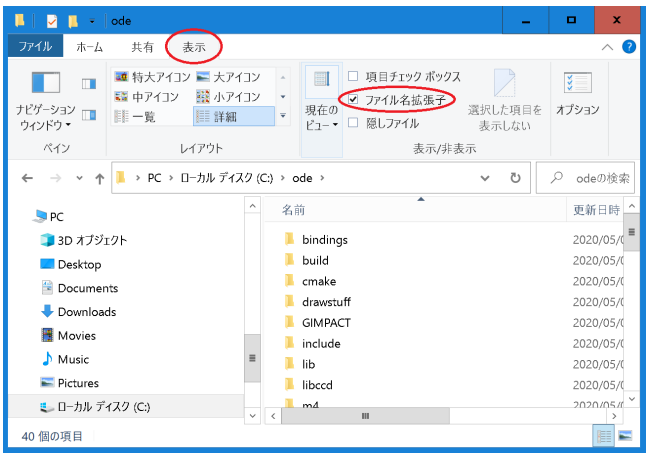
）で C:¥に移動し，「貼り付け」る．
（結果として，右下に示すように，C ドライブの直下に ode フォルダが存在するようになる）

⑦ 以降の説明の（**ODE のフォルダ**）というのは C:¥ode のこととなる．





※補足
エクスプローラの初期設定では，拡張子が表示されていないため，以降の作業のために表示する設定に変更しておく．１－３のQ&Aの2つ目を参照．エクスプローラの上の方にある「表示」メニューを選択し，「ファイル名拡張子」を☑する，これにより，slnやexeなどの拡張子が表示されるようになる．



（4）　ODE インストーラの実行：

コマンドプロンプトで，ディレクトリを（ODE のフォルダ）¥build へ移動し，以下のように premake4 を実行する．

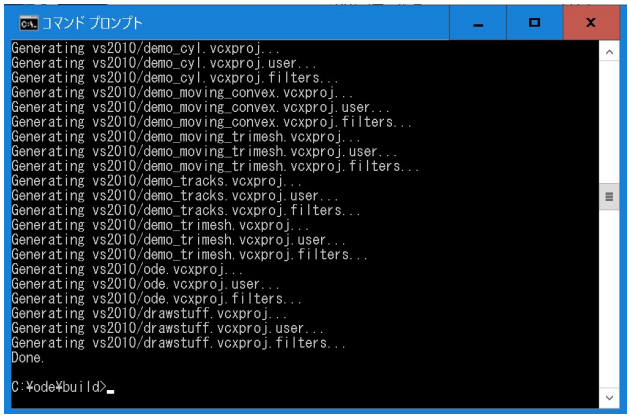
Windows のスタートメニューで“cmd”と入力して起動（以下のコマンド（スペースを気を付けること）を入力）

```
C:¥User¥user> cd ¥ode¥build
C:¥ode¥build> premake4 --only-double --only-static --with-demos --with-libccd vs2010
```

↑　↑　↑　↑　↑

矢印のところはスペースが入ります

多数のメッセージの表示ののち，終了するので，コマンドプロンプトを閉じる．



（5）　ODE のセットアップ

Visual Studio を起動し，（ODE のフォルダ）¥build¥vs2010¥ode.sln を開く．

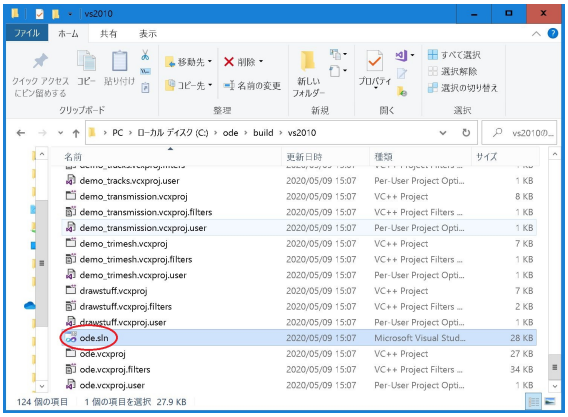
「Visual Studio 変換ウィザードへようこそ」ウインドウが開き，バックアップが必要ないので「いいえ(O)」にチェックを入れ，「次へ(N)」をクリックする．

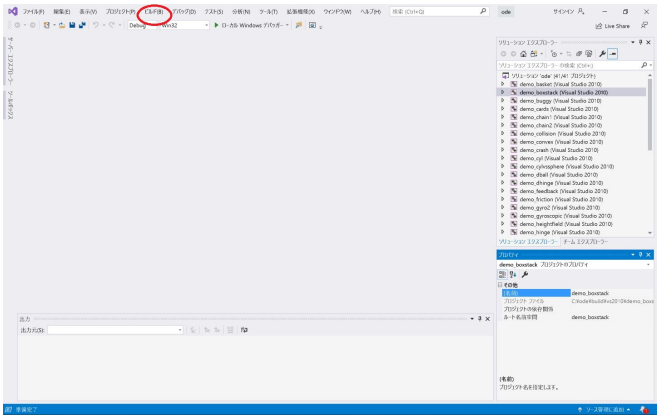
「変換準備完了」の画面になるので「完了(F)」をクリック．

「変換の完了」の画面になるので「閉じる」をクリック．

Visual Studio のメニューバーから「デバッグ(D)」→「ソリューションのビルド(B)」を選択し，ビルドを完了させる．

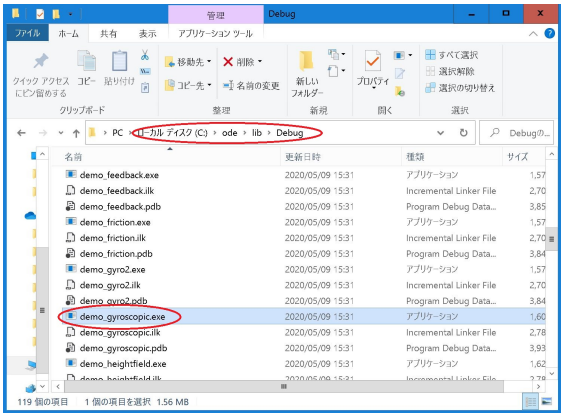
ただし，Visual Studio 2019 では以下のように開け，メニューバーから「ビルド(B)」→「ソリューションのビルド(B)」を選択し，ビルドを完了させる．なお，v100 に関するエラーが出る場合は，メニューバーから「プロジェクト(P)」→「ソリューションの再ターゲット」を実行してから，ビルドを行う．





（6）　ODE のインストール確認（デモプログラムの実行）

デモプログラムの実行ファイルが（ODE のフォルダ）¥lib¥Debug に生成される．その中の demo_... .exe（たとえば demo_gyroscopic.exe）をダブルクリックし，起動すれば ODE のインストールが完了したことが確認できる．



【参考】

・Visual Studio でデバッグするときはウィンドウ上部の「ソリューションの構成」を「Debug」，「ソリューションプラットフォーム」を「x86」に設定する．

・PC のスペック，OS のバージョン，開発環境などの条件によって，プログラムの速度，計算値の精度，コンパイル時の警告・エラーなどの挙動がだいぶ変わってしまう可能性があるので，注意すること．

・参考となるWebサイト

https://kayastyle.jp/install-visualstudio-community-2015/
https://imagingsolution.net/program/visualstudio/download-old-version-visual-studio/
https://demura.net/robot/9ode/9928.html
http://hara.jpn.com/_default/ja/Software/Build_ODE_with_MSVC.html
https://qiita.com/ymsk_sky/items/ef71a6b00ae719c7ae6d

・ディレクトリ構成

LocoRoboSim

—	Debug	ビルド時に生成されたデバッグ実行ファイルを含むフォルダ
|	— LocoRoboSim.exe	
|	— LocoRoboSim.ilk	
|	└── LocoRoboSim.pdb	
—	LocoRoboSim	
|	— Debug	ビルドログを記したデバッグファイルのフォルダ
|	— LocoRoboSim.vcxproj	VisualStudio のプロジェクトファイル
|	— LocoRoboSim.vcxproj.filters	
|	— LocoRoboSim.vcxproj.user	
|	— drawstuff	ODE の drawstuff のソースファイルフォルダ
|	— drawstuffd.dll	drawstuff の dll ファイル
|	— include	ODE の include ファイルフォルダ
|	— lib	ODE の lib ファイルフォルダ
|	— Loco-data.txt	姿勢制御読み込み txt ファイル
|	— main.cpp	C++のソースコード
|	└── ode_doubled.dll	drawstuff の dll ファイル
—	LocoRoboSim.VC.db	Visual Studio のコンバージョン時に発生したファイル
└──	LocoRoboSim.sln	Visual Studio のソリューションファイル

1−2. 開発環境が GCC の場合

（1）　GCC のインストール

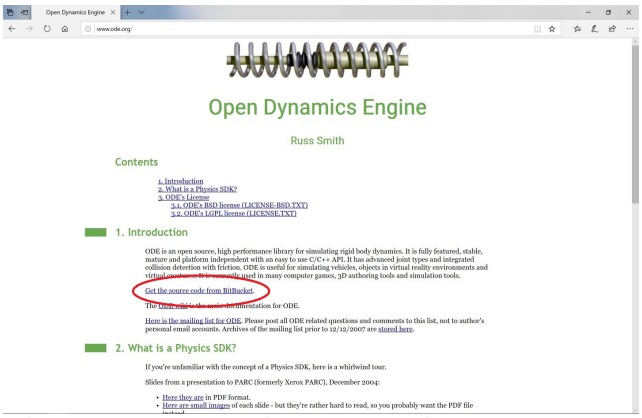
MinGWをインストールする．
以下のURLから，MinGWGetインストーラのセットアップファイルmingw-get-setup.exeをダウンロードする
http://www.mingw.org/ → https://osdn.net/projects/mingw/releases/68260
（2020年　Ver.0.6.3beta）
mingw-get-setup.exeを起動し，MinGWGetインストーラをインストールする．
デフォルトではMinGWはCドライブにインストールされる
MinGWGetインストーラを起動し，「Basic Setup」より，
「mingw-development-toolkit-bin」，「mingw32-base-bin」，「mingw32-gcc-g++-bin」，「msys-base-bin」の各項目のチェックボックスをクリックし，　表示されたコンテキストメニューで,「Mark for Installation」を選択し，上部メニューバーの「Installation」→「Apply Changes」でインストールをする．
コンピュータ→システムのプロパティ→システムの詳細設定→環境変数→システム環境変数→Pathに，（MinGWフォルダ名）¥binを追加する．
コマンドプロンプトから，gcc --helpで，インストールが完了できたかを確認できる．

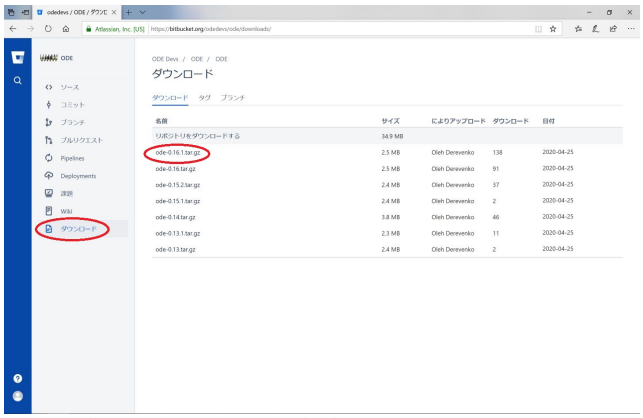
（2）　make 環境の構築

1.にて，「msys-base-bin」　でmake環境もインストールされる．
コンピュータ→システムのプロパティ→システムの詳細設定→環境変数→システム環境変数→Pathに，（MinGWフォルダ名）¥msys¥1.0¥binを追加する．
コマンドプロンプトから，make --helpで，インストールが完了できたかを確認できる．

（3）　Open Dynamics Engine (ODE)のダウンロード

Open Dynamics Engine (ODE)のソースファイルを以下のURLからダウンロードし，ディスク直下に解凍する．
http://www.ode.org/ → https://bitbucket.org/odedevs/ode/downloads/





（2014年　Ver.0.13（Russ Smith製）　2020年　Ver.0.16.1（Oleh Derevenko製））
例：Cドライブ直下に解凍した時C:¥ode
*.tar.gzファイルは，たとえばフリーソフトの7-Zip（https://sevenzip.osdn.jp/）や
Lhaplus（https://forest.watch.impress.co.jp/library/software/lhaplus/）などで解凍可能．

（4）　インストーラの実行

コマンドプロンプトで，ディレクトリを（ODEフォルダ名）¥buildへ移動し，以下のようにpremake4を実行する．

```
> cd ¥ode¥build
> premake4 --cc=gcc --only-double --only-static --os=windows --platform=x32 --with-demos --with-libccd gmake
```

または

```
> cd ¥ode¥build
> premake4 --cc=gcc --only-double --only-static --os=windows --platform=x64 --with-demos
--with-libccd gmake
```

（5） make ファイルの実行

コマンドプロンプトで，ディレクトリを（ODEフォルダ名）¥build¥gmakeへ移動し，以下のようにMakefileを実行する．

```
> cd ¥ode¥build¥gmake
> make -f Makefile CC=gcc
```

（6） インストール完了の確認

（ODEフォルダ名）¥lib¥Debug内に，.exeファイルとliboded.a，libdrawstuffd.aファイルが出力される．その中のdemo_... .exeをダブルクリックし，起動すればODEのインストールが完了したことが確認できる．

（7） ODE 関係ファイルのコピー

（ODEフォルダ名）¥lib¥Debug内の，liboded.a，libdrawstuffd.aを，MinGWフォルダ内の，（MinGWフォルダ名）¥libフォルダ及び（MinGWフォルダ名）¥mingw32¥libフォルダ，（MinGWフォルダ名）¥lib¥gcc¥mingw32¥9.2.0へコピーする．

（8） glut の導入

今回導入するのは後継版のfreegluとする．

http://www.transmissionzero.co.uk/software/freeglut-devel/ →「Download freeglut 3.0.0 for MinGW」(Ver.3.0.0)

Zipファイル内の

¥freeglut¥lib内のlibfreeglut.a，libfreeglut_static.aを，MinGWフォルダ内の，（MinGWフォルダ名）¥libフォルダ及び（MinGWフォルダ名）¥mingw32¥libフォルダ，（MinGWフォルダ名）¥lib¥gcc¥mingw32¥9.2.0へコピーする．

¥freeglut¥include¥GLフォルダを，MinGW内の，（MinGWフォルダ名）¥includeフォルダ及び（MinGWフォルダ名）¥lib¥gcc¥mingw32¥9.2.0¥includeフォルダへコピーする．

¥freeglut¥bin内のfreeglu.dll（x64用もある）をC:¥WINDOWS¥systemフォルダおよびsystem32へコピーする．

（9） resources.o ファイルの作成

コマンドプロンプトで，ディレクトリを（ODEフォルダ）¥drawstuff¥srcへ移動し，

```
> cd ¥ode¥drawstuff¥src
> windres resources.rc -o resources.o
```

と実行し，resources.oファイルを出力する．

（10） 環境変数の追加

コンピュータ→システムのプロパティ→システムの詳細設定→環境変数→システム環境変数→Pathに，（ODEフォルダ名）¥lib¥Debugを追加する．

（11） コンパイルのための準備

ODEフォルダ内に適当に自分のソースコードを置くディレクトリを作り，ODEのソースコードをプログラミングする．

例：ディレクトリを”myprog1”としたとき，C:¥ode¥myprog1¥hello.cpp

（12） コンパイルの実行

コマンドプロンプトで，（１１）のフォルダ内に移動し，以下のようにソースコードをコンパイルする．

g++ -IC:¥(ODEフォルダ名)¥include/ -LC:¥ (ODEフォルダ名)¥lib¥Debug/ -o 出力ファイル名 ソースコ

ード名.cpp -loded -ldrawstuffd -lfreeglut -lglu32 -lopengl32 -lwinmm -lgdi32 C:¥ (ODEフォルダ名)¥drawstuff¥src¥resources.o

例：Cドライブ直下にODEのファイルがある時

```
> g++ -IC:¥ode¥include/ -LC:¥ode¥lib¥Debug/ -o hello hello.cpp -loded -ldrawstuffd -lfreeglut -lglu32 -lopengl32 -lwinmm -lgdi32 C:¥ode¥drawstuff¥src¥resources.o
```

【参考】
・環境設定が完了すれば，フォルダ内にはソースコードファイルのみで，コンパイルをすれば実行ファイルが生成される．
・Webサイト
https://algorithm.joho.info/programming/c-language/windows10-mingw-gcc-install/
https://blog.t-semi.org/?p=44

1−3． Q&A

Q1) can't open image file　～　sky.ppm'のエラーがでる．
A1) drawstuff フォルダのパスを確認する
　　フォルダがスクリプトと同じ階層にがあるなら"drawstuff/textures"
　　2 階層前なら"../../drawstuff/textures"
と，ソースコードに記述する．

Q2)作成したり出力したりしたソースコードや実行ファイルが見つからない．
A2) Windows の標準の設定では，拡張子の表示がオフになっています．以下の方法で，拡張子を表示する設定にしてください．
Windows7 の場合：エクスプローラーのウィンドウの右上の「整理」メニューから，「フォルダーと検索のオプション」を選択し，「表示」タブを選択し，「詳細設定」リスト内の，「登録されている拡張子は表示しない」のチェックボックスをオフにする．
Windows10 の場合：エクスプローラーのウィンドウの上部メニュータブの，「表示」タブから「ファイル名拡張子」のチェックボックスをオンにする．

2. Mac 用セットアップ

※Terminal（ターミナル）は，アプリケーション/ユーティリティの中にある Mac 用アプリである．Linux と同等のコマンドを使うことができる．なお、ソースコードの編集には、Xcode を用いるため、事前に Xcode をダウンロードしておくこと。その際、App Store から App をインストールするために、Apple ID が必要となる。

2−1. ターミナルを用いた実行方法（基礎）

最低限の基本的なターミナルにおけるコマンドは以下の通りである．また，tab キーを使うことで補完機能が使える（文字を途中まで打って，tab キーを 2 回押すと候補が表示される）．

（基本，以下の ls と cd を使う）

・ls:　ディレクトリの中身を表示（list directory）

例：naoyukikubota@NaoyukinoMacBook-Air ~ % ls

　　Desktop　　Music　　Documents　　…

・cd:　ディレクトリを移動(change directory)

　cd Desktop →　デスクトップへディレクトリを移動

　cd ../　　→　一つ上のディレクトリへ戻る

　cd ~　　→　自分の home directory（Finder 左リストにて自分の名前が書かれた家のアイコンで表されるフォルダ）に移動する．

　例：ユーザー名@MacBook-Air ~ % cd ..

　　ユーザー名@MacBook-Air /Users % cd ~

　　ユーザー名@MacBook-Air ~ % cd Documents

　　ユーザー名@MacBook-Air Documents %

・mkdir:　ディレクトリを作る．（Mac 上でフォルダを作成してもよい, make directory）

　例： mkdir sample → sample というディレクトリを作る

・tar アーカイブ（圧縮）ファイルの操作を行う．オプションをつけて，圧縮ファイルの解凍などを行う

　ことができる(tape file archiver：昔は、磁気テープを用いて保存されていた).

　例： tar xzf xxx.tar →指定した圧縮ファイルxxx.tarを解凍する

・cp:　ファイルのコピーを行う．オプションをつけて，コピー先の指定などを行うことができる．(copy)

　例：cp xxx.c sample → ファイル名xxx.cをディレクトリsampleの中にコピー

・sudo: 「スーパーユーザー（rootユーザー,管理者）」の権限が必要なコマンドをsudoコマンド経由で実行させる．（superuser do）

　例：sudo cp …

・make: フォルダ内にあるmakeファイルを実行する．

【参考】

http://www.ysr.net.it-chiba.ac.jp/data/unix2.html

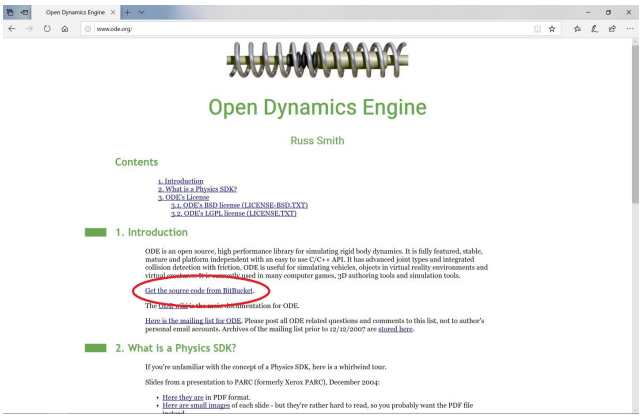
https://hydrocul.github.io/wiki/commands/

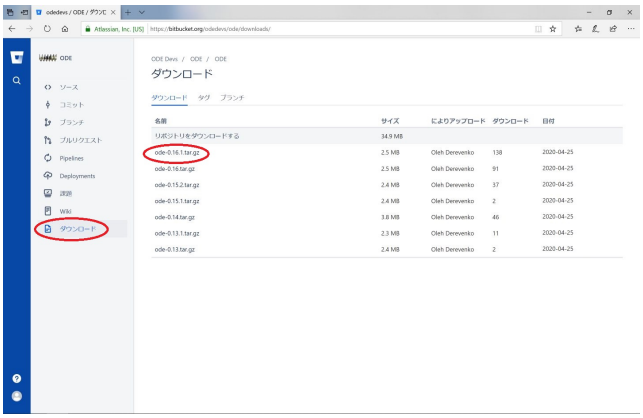
2−2.　開発環境が Xcode の場合

（1）　Open Dynamics Engine (ODE)のダウンロード

Open Dynamics Engine (ODE)のソースファイルを以下のURLからダウンロードし，ダウンロードフォルダ内でもいいのでode-(version名).tar.gzを解凍する．

http://www.ode.org/ → https://bitbucket.org/odedevs/ode/downloads/





（2014年　Ver.0.13（Russ Smith製）　2020年　Ver.0.16.1（Oleh Derevenko製））

```
tar xzf ode-(version名).tar.gz
```

（2）　対象ディレクトリへの移動

ターミナルを起動して，解凍した ODE のディレクトリへ移動する．

```
cd ode-(version名)
```

（3）　Makefile の作成準備

次のように入力し，スクリプトファイルを実行し，Makefile を作成する．

```
./configure --enable-double-precision
```

（4）　Makefile の作成

次のように入力し，Makefile に基づいてアプリのコンパイルを行う．

```
make
```

（5）　ODE のインストール

次のように入力し，コンパイルされたアプリをインストールする．

```
sudo make install
```

（6）　ODE 関係ファイルのコピー（1）

インストールされた ODE に関するフォルダは/usr/local/に追加される．drawstuff に関するファイルを追加するため，次のように入力し，drawstuff のインクルードをコピーする．

```
cd include
sudo cp -r drawstuff /usr/local/include/
```

（7）　ODE 関係ファイルのコピー（2）

次のように入力し，drawstuff ライブラリをコピーする．

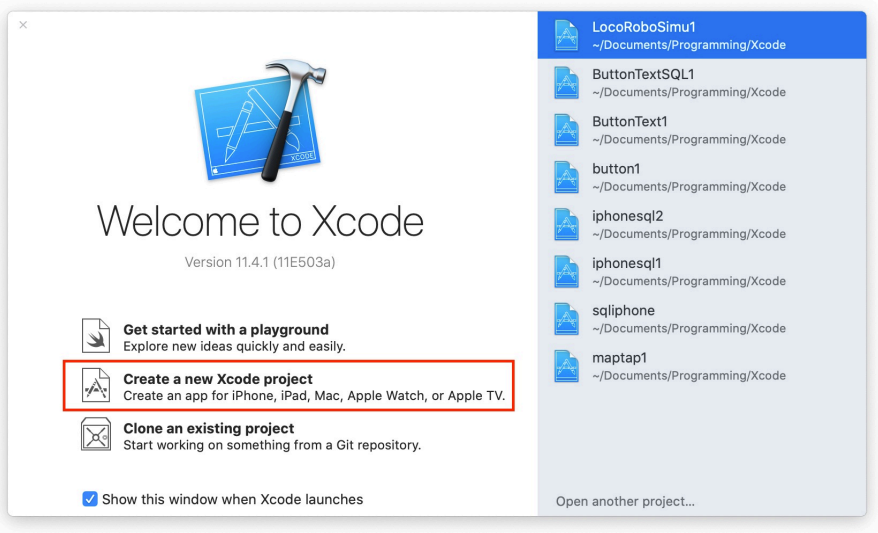
```
cd ../drawstuff/src/.libs
sudo cp * /usr/local/lib/
```

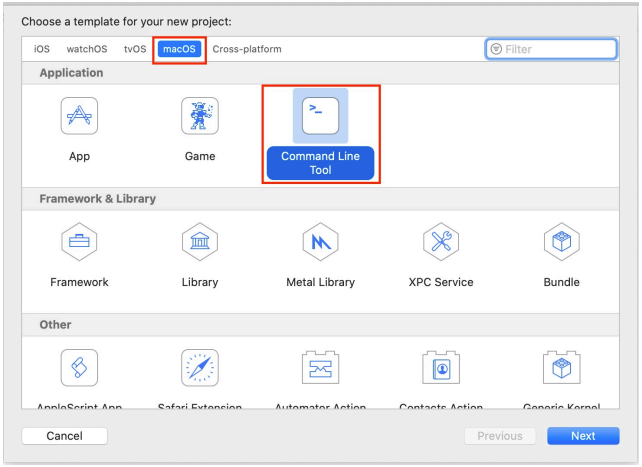
（8）　Xcode のインストール

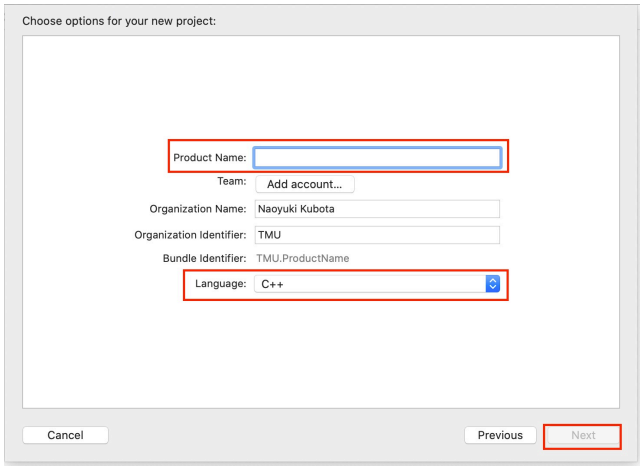
App Store を起動し，「Xcode」と検索し，「Xcode」をインストールする．ただし，App Store から App をインストールする際には，Apple ID が必要となる．

（9）　Xcode の起動

Xcode を起動し，「Create a new Xcode project」を選択し，「macOSX」タブより，「Command Line Tool」を選択し，「Next」をクリックする．「Product Name」に英数字でプロジェクト名を記入し，「Language」を「C++」に選択し，「Next」を選択する．



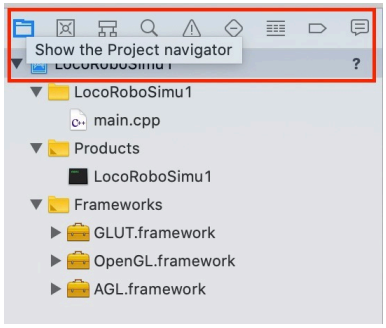




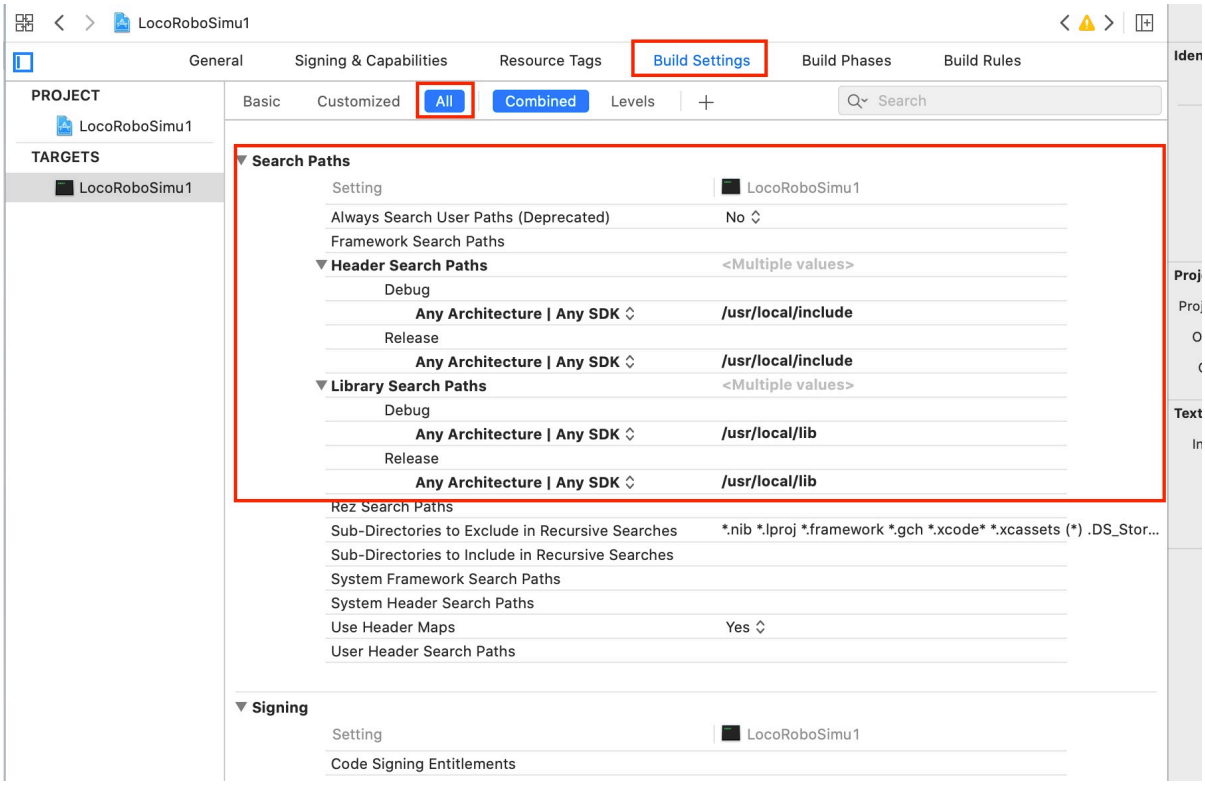
（10）　Xcode プロジェクトファイルの設定

上記の ODE インストール方法を実施の上，Xcode プロジェクトに以下を設定する．

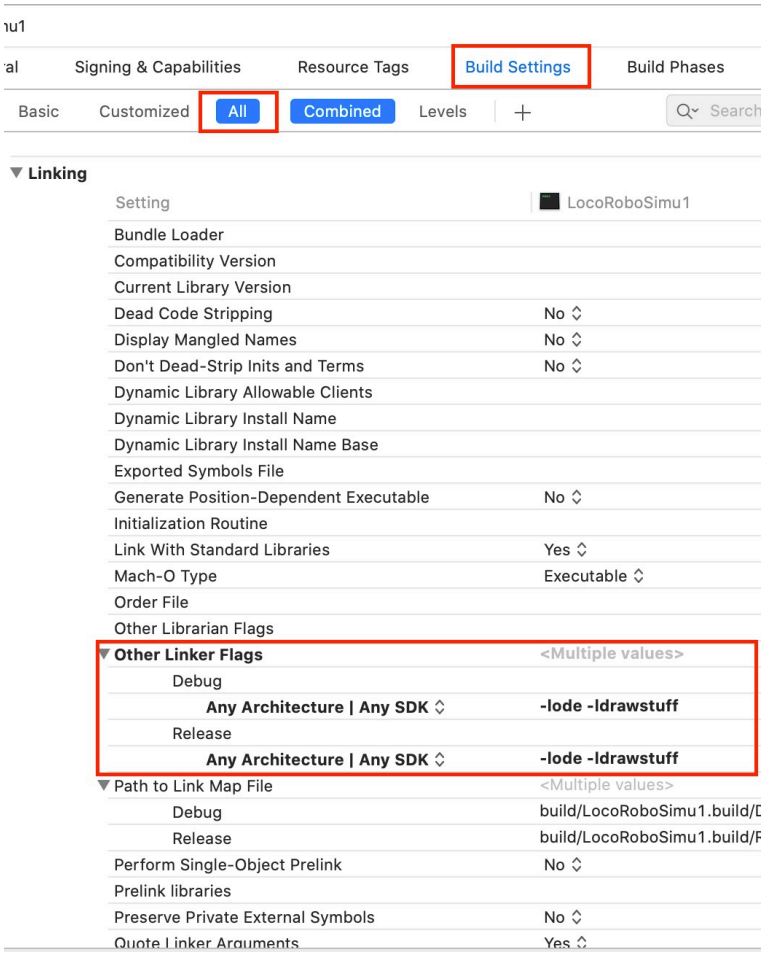
・画面左側のプロジェクト名をクリックして Project Navigator を開き，



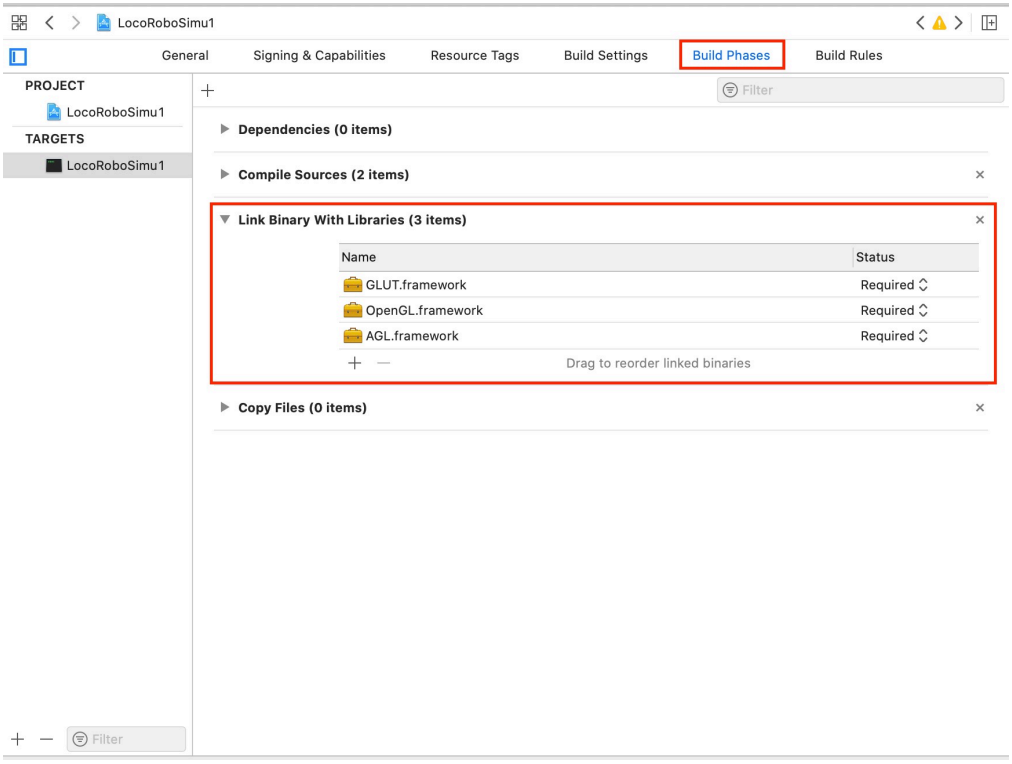
TARGETS の Build Settings,All,Search Paths の Header Search Paths に/usr/local/include を追加
TARGETS の Build Settings,All,Search Paths の Library Search Paths に/usr/local/lib を追加

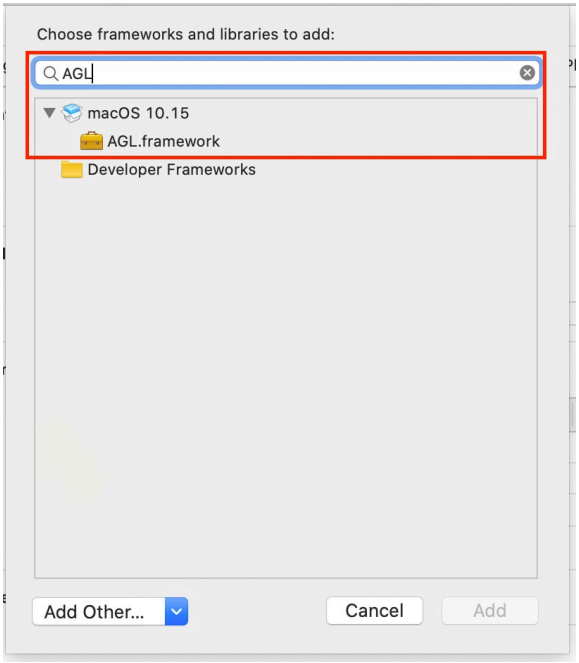


TARGETS の Build Settings,All,Linking の Other Linker Flags に-lode と-ldrawstuff を追加

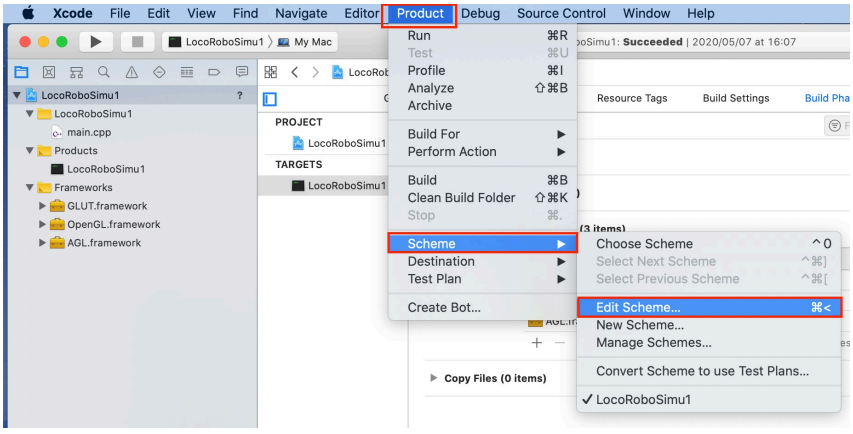


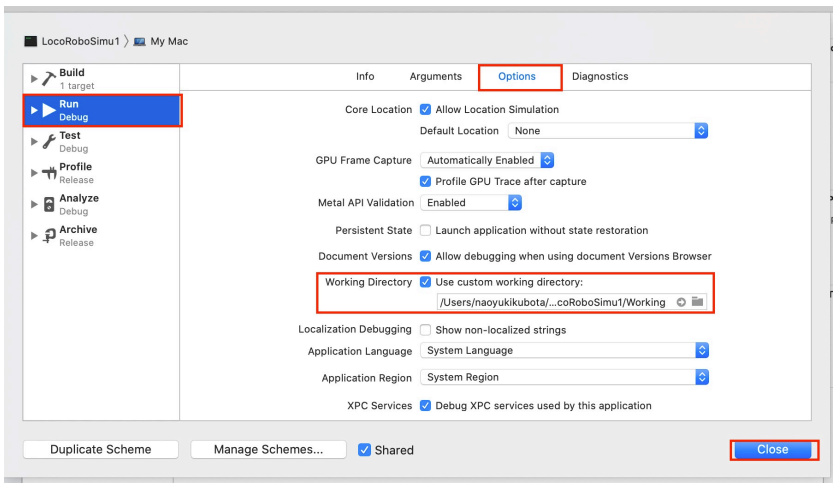
TARGETS の Build Phases の Link Binary with Libraries に AGL.framework, OpenGL.framework, GLUT.framework を追加（「＋」ボタンを選択し，検索ボックスに各々の名称を入力し検索して選択）する．





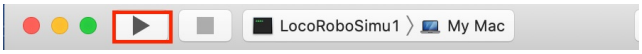
・画面上側のメニューバーの Product→Scheme→Edit Scheme…を選択後,
　Run の Options の Working Directory の Use custom working directory にチェックを入れ, プロジェクトのソースフォルダ内の作業ディレクトリ（textures フォルダと, 歩行動作データ（Loco-data.csv, Loco-data.txt など）のあるディレクトリ）を指定する. 作業ディレクトリには, ode-(version 名)/drawstuff からコピーした textures フォルダと, 歩行動作データ (Loco-data.csv, Loco-data.txt など) を入れておく. (【参考】のディレクトリ構成を参照のこと)

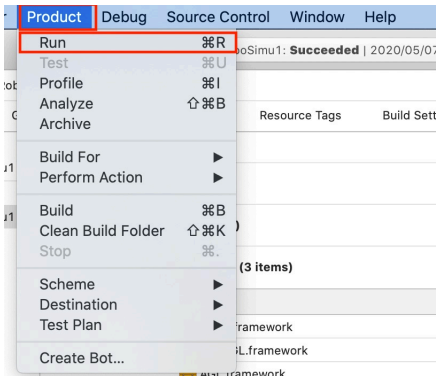




（11） 実行方法（コンパイルと実行）

ソースコードを編集後，ウィンドウ右上の Run ボタン（▶マーク）を選択するか，メニューバーの「Product」→「Run」（Command⌘＋r）を選択し，ビルドと実行を行う．「Build Succeeded」と画面に一瞬表示されれば，プログラムのビルドが成功し，プログラムが実行される．「Build Failed」と画面に一瞬表示された場合は，プログラムのビルドが失敗し，プログラムが実行されないため，再度プロジェクトの設定やソースコードを見直すこと．

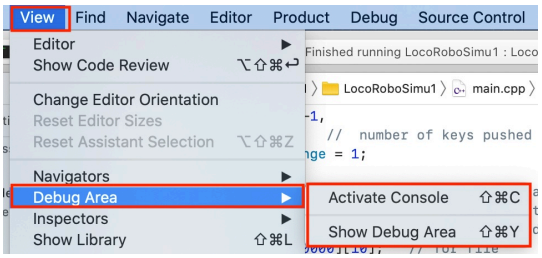


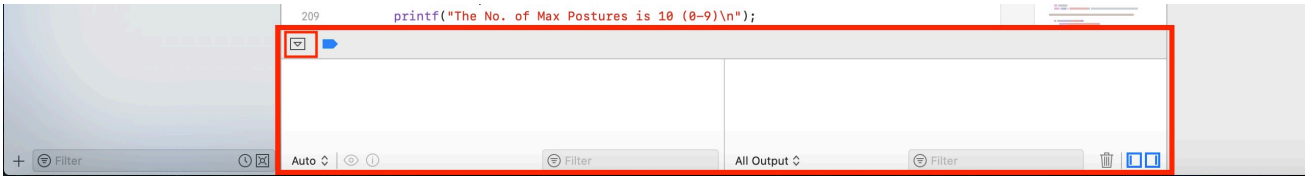


ビルドが成功しプログラムが実行される際，「”（プロジェクト名）”から”書類”フォルダ内のファイルにアクセスしようとしています。」というウィンドウが表示された場合は，「OK」を選択すればプログラムの実行は続行される．

実行しているプログラムの強制終了は，ウィンドウ右上の Stop ボタン（■マーク）を選択するか，メニューバーの「Product」→「Stop」（Command⌘+.）を選択，または ODE のシミュレータウインドウの画面上で「[Command⌘]+[q]」（コマンド⌘を押しながら，q のキーを押す）である．

デバッグメッセージや出力メッセージが見たい場合は，ウィンドウ下部の表示ボタン（四角の中に三角があるボタン，図参照）をクリックするか，メニューバーの「View」→「Debug Area」→「Activate Console」（Shift+Command⌘+c）または「Show Debug Area」（Shift+Command⌘+y）を選択することで，ウィンドウ下部にデバッグメッセージや出力メッセージを表示するデバッグエリアが表示される．





【参考】

・参考となる Web サイト

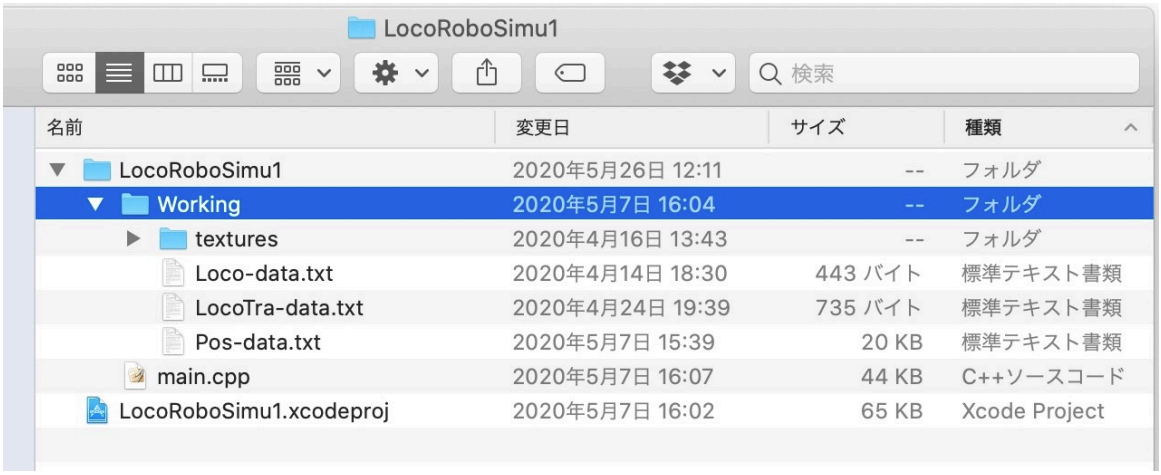
https://studio-kikoro.appspot.com/blog/20141130_ode.html
https://w.atwiki.jp/kencyo/pages/14.html
http://seiya-kumada.blogspot.com/2012/09/open-dynamics-enginemac.html
https://rage.at.webry.info/200909/article_5.html

・ディレクトリ構成

※（プロジェクト名は任意でよい．ここではプロジェクト名を「LocoRoboSim」としている．）

LocoRoboSim

|	LocoRoboSim		
—	|	Working	参照する作業フォルダ
—	—	|	Loco-data.txt	姿勢制御読み込み txt ファイル
—	—	└—	textures	drawstuff からの textures 参照フォルダ
—	└—	main.cpp	C++のソースコード
└—	LocoRoboSim.xcodeproj		Xcode のプロジェクトファイル



2−3. 開発環境がターミナルの場合

（1） Open Dynamics Engine (ODE)の設定

「２－２．開発環境が Xcode の場合」の（１）から（７）までを行う．

（2） 作業ディレクトリの作成

自分の home directory や書類フォルダなど，どこでも良いので，ODE のソースコードなどを置く作業ディレクトリを作成する．そのフォルダの中に，ode-(version 名)/drawstuff からコピーした textures フォルダと，歩行動作データ（Loco-data.csv, Loco-data.txt など）を置く．

（3） ターミナル上でのコンパイル方法

ターミナル上でのコンパイルは、以下のように行う。

```
g++ -I/usr/local/include -L/usr/local/lib -lode -ldrawstuff -lm -framework GLUT -framework OpenGL -o a.out
```

の後に，以下の例のようにファイル名（main.cpp など）を入力する．

```
g++ -I/usr/local/include -L/usr/local/lib -lode -ldrawstuff -lm -framework GLUT -framework OpenGL -o a.out main.cpp
```

※”a.out”は，（英数字名）.out で任意のファイル名を作っても良い．

（4） 実行方法

コンパイルされたプログラムの実行方法は， ./a.out （./（英数字名）.out)で実行可能である．実行しているプログラムの強制終了は，terminal 画面上で，「[Ctrl] + [c]」（コントロールを押しながら，c のキーを押す）または，ODE のシミュレータウインドウの画面上で「[Command]+[q]」（コマンド⌘を押しながら，q のキーを押す）である．

```
./a.out
```

2—4. Q&A

Q1) MacOSXにて，コンパイルし，作成されたプログラムを実行すると，シミュレータがウィンドウの左下に小さく表示されてしまう．

A1) MacOSXのGLUTにバグがあるようです．小さい画面のままで行ってください．

Q2) 2-2.(3)を実行したところ，「 xcode-select: note: no developer tools were found at '/Applications/Xcode.app', requesting install. Choose an option in the dialog to download the command line developer tools.」と表示されて処理が中断されてしまう．

A2) XcodeをAppStoreからインストールをして，再度実行をしてください．

【参考】

・Xcode や ODE がインストールされていない環境でも，a.out 他をコピーすれば，terminal 上から実行可能である．

・Web サイト

https://studio-kikoro.appspot.com/blog/20141130_ode.html
https://w.atwiki.jp/kencyo/pages/14.html
http://seiya-kumada.blogspot.com/2012/09/open-dynamics-enginemac.html
https://rage.at.webry.info/200909/article_5.html

3. Linux 用セットアップ（参考）

Linux(Deepin Linux v20 beta based on Debian10)におけるインストール方法について述べる．

3−1. ターミナルを用いた実行方法（基礎）

最低限の基本的なターミナルにおけるコマンドは以下の通りである．

・ls:　ディレクトリの中身を表示（list directory）

例：naoyukikubota@NaoyukinoMacBook-Air ~ % ls

　　Desktop　　Music　　Documents　…

・cd:　ディレクトリを移動(change directory)

　cd Desktop →　デスクトップへディレクトリを移動

　　cd ../　　→　一つ上のディレクトリへ戻る

　　cd ~　　→　自分の home directory（Finder 左リストにて自分の名前が書かれた家のアイコンで表されるフォルダ）に移動する．

　例：

　ユーザー名@MacBook-Air ~ % cd ..

　ユーザー名@MacBook-Air /Users % cd ~

　ユーザー名@MacBook-Air ~ % cd Documents

　ユーザー名@MacBook-Air Documents %

・mkdir:　ディレクトリを作る．（Mac 上でフォルダを作成してもよい, make directory）

　例：　mkdir sample　→sample というディレクトリを作る

・tar アーカイブ（圧縮）ファイルの操作を行う．オプションをつけて，圧縮ファイルの解凍などを行うことができる(tape file archiver：昔は、磁気テープを用いて保存されていた).

　例：指定した圧縮ファイルを解凍するとき

　　tar xzf　xxx.tar

・cp:　ファイルのコピーを行う．オプションをつけて，コピー先の指定などを行うことができる．(copy)

　例：cp　ファイル名

・sudo:　「スーパーユーザー（rootユーザー,管理者）」の権限が必要なコマンドをsudoコマンド経由で実行させる．（superuser do）

　例：sudo cp …

・make: フォルダ内にあるmakeファイルを実行する．

【参考】

http://www.ysr.net.it-chiba.ac.jp/data/unix2.html

https://hydrocul.github.io/wiki/commands/

（1） パッケージのインストール

ターミナルを起動して，以下のコマンドを入力し，X11 のパッケージと GLU，glut のパッケージをインストールする．

```
sudo apt-get update
sudo apt-get install libgl1-mesa-dev libx11-dev libglu1-mesa-dev
```

または

```
sudo apt-get update
sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev
```

（2） Open Dynamics Engine (ODE)の設定

「２－２．開発環境が Xcode の場合」の（１）から（５）までを行う．

（3） 作業フォルダの作成

ODEを解凍したフォルダの一つ下のフォルダへ移動し，作業フォルダを作成する（フォルダ名は英字ならば任意の名前でよい）

```
cd ..
mkdir (作業フォルダ名)
```

（4） ODE 関連ファイルのコピー

作業フォルダ内に，/ODE/drawstuff/textures をコピーする

```
cd ode-(version 名)
cd drawstuff
cp -r drawstuff (作業フォルダ名)
```

（5） コンパイル方法

作業フォルダ内で，ソースコードを作成し，以下のように入力し，コンパイルを行う．

```
g++ -DHAVE_CONFIG_H -I. -I~/ (ODE フォルダパス)/ode/src -I~/ (ODE フォルダパス)/include -I~/(ODE フォルダパス)/include -DDRAWSTUFF_TEXTURE_PATH="¥""~/(作業フォルダパス)/textures¥"" -DdTRIMESH_ENABLED -g -O2 -MT （出力ファイル名）.o -MD -MP -MF .deps/（出力ファイル名）.Tpo -c -o （出力ファイル名）.o （ソースコードファイル名）.cpp

g++ -g -O2 -o （出力ファイル名）（出力ファイル名).o ~/ (ODE フォルダパス)/drawstuff/src/.libs/libdrawstuff.a -lX11 ~/ (ODE フォルダパス)/ode/src/.libs/libode.a -lGLU -lGL -lrt -lm -lpthread
```

例：

```
g++ -DHAVE_CONFIG_H -I. -I~/Downloads/ode-0.16.1/ode/src -I~/Downloads/ode-0.16.1/include -I~/Downloads/ode-0.16.1/include -DDRAWSTUFF_TEXTURE_PATH="¥""~/Downloads/new/textures¥"" -DdTRIMESH_ENABLED -g -O2 -MT mainEX08.o -MD -MP -MF .deps/mainEX08.Tpo -c -o mainEX08.o mainEX08.cpp

g++ -g -O2 -o mainEX08 mainEX08.o ~/Downloads/ode-0.16.1/drawstuff/src/.libs/libdrawstuff.a -lX11 ~/Downloads/ode-0.16.1/ode/src/.libs/libode.a -lGLU -lGL -lrt -lm -lpthread
```