



◆CygwinでDockerを使う
 ◆Unity開発でのXcodeのバージョンアップ
 ◆Re:VIEWのClをAWS CodeBuildで
 ◆WebAssemblyで行列演算
 ◆Pythonインスタンスの属性の実装について



はじめに

このたびは本書をお手に取っていただきありがとうございます。

本書は、KLab 株式会社の有志にて作成された KLab Tech Book の第2弾です。今回 もさまざまな内容が集まりました。Cygwin 環境で Docker を使う方法・Unity プロジェ クトでの Xcode のバージョンアップ・AWS CodeBuild・WebAssembly で行列計算・ CPython のインスタンス属性となっています。

内容は各章ごとに独立していますので、どの順番からお読みくださっても問題ありま せん。

楽しんでいただければさいわいです。

岡本和樹

お問い合わせ先

本書に関するお問い合わせは tech-book@support.klab.com まで。

免責事項

本書に記載された内容は、情報の提供のみを目的としています。したがって、本書を用 いた開発、製作、運用は、必ずご自身の責任と判断によって行ってください。これらの情 報による開発、製作、運用の結果について、著者はいかなる責任も負いません。

目次

はじめに		2
お問い	合わせ先	2
免責事	項	2
第1章	Cygwin 環境で Docker を快適に使うために	5
1.1	はじめに	5
1.2	Cygwin で Docker を使うときの問題点	5
1.3	TTY モードを有効にできない問題を回避する	6
1.4	Cygwin 環境のパスをそのまま指定できるようにする........	8
1.5	まとめ	11
第2章	Unity で開発しているスマホゲームで Xcode のバージョンアップをする	12
2.1	Unity で開発しているスマホゲームのビルド手法	12
2.2	iOS アプリのビルド手法	15
2.3	Xcode バージョンアップのケーススタディ	22
2.4	Xcode バージョンアップによるビルド手法の変更点、キャッチアップ方法	29
2.5	おわりに	31
第3章	Re:VIEW で書いた原稿の CI を AWS CodeBuild で回す	32
3.1	ことのはじまり	32
3.2	CI 環境、フロー	32
3.3	AWS CodeBuild とは	35
3.4	GHE / AWS (S3, CodeBuild, ECS) を利用した Re:VIEW CI 環境構築	36
3.5	CI 環境構築でハマったところ	56
3.6	おわりに	58
第4章	WebAssembly で行列の演算をする	59
4.1	はじめに	59
4.2	4 行 4 列行列の乗算	60
4.3	JavaScript による実装.......................	60

4.4	Emscripten による実装	61
4.5	ベンチマーク	64
4.6	まとめ	65
第5章	Python インスタンスの属性は辞書(dict)で管理されているのか調べて	
	みた	66
5.1	dis - バイトコードの逆アセンブラ	67
5.2	ceval.c - バイトコードの実行処理	67
5.3	object.c - PyObject_GetAttr の実装	68
5.4	おわりに	70
第6章	リアルの街を Unity に取り込む ~GIS へのいざない~	71
61	帝田 生会車別など	71
0.1	間用、元1]事例なと	11
$6.1 \\ 6.2$	間用、元1事例など 自前でデータを用意する意味	71 73
6.2 6.3	 間用、元1事内など 自前でデータを用意する意味 地理情報の基礎知識 	73 73
6.1 6.2 6.3 6.4	 間用、元1事内など 自前でデータを用意する意味 地理情報の基礎知識 QGIS のインストール 	71 73 73 77
$ \begin{array}{c} 0.1 \\ 6.2 \\ 6.3 \\ 6.4 \\ 6.5 \\ \end{array} $	 商用、元打事内など 自前でデータを用意する意味 地理情報の基礎知識 QGIS のインストール データのダウンロード 	71 73 73 77 77 78
$6.1 \\ 6.2 \\ 6.3 \\ 6.4 \\ 6.5 \\ 6.6$	商用、先行事例など 1 自前でデータを用意する意味 1 地理情報の基礎知識 1 QGIS のインストール 1 データのダウンロード 1 データの変換 1	73 73 77 78 85
$ \begin{array}{c} 6.1 \\ 6.2 \\ 6.3 \\ 6.4 \\ 6.5 \\ 6.6 \\ 6.7 \\ \end{array} $	商用、元打事内なと 1 自前でデータを用意する意味 1 地理情報の基礎知識 1 QGIS のインストール 1 データのダウンロード 1 データの変換 1 データの読み込みと QGIS の基本操作 1	71 73 73 77 78 85 90
$\begin{array}{c} 6.1 \\ 6.2 \\ 6.3 \\ 6.4 \\ 6.5 \\ 6.6 \\ 6.7 \\ 6.8 \end{array}$	間用、先行事例など 1 自前でデータを用意する意味 1 地理情報の基礎知識 1 QGIS のインストール 1 データのダウンロード 1 データの読み込みと QGIS の基本操作 1	71 73 73 77 78 85 90 92
$\begin{array}{c} 6.1 \\ 6.2 \\ 6.3 \\ 6.4 \\ 6.5 \\ 6.6 \\ 6.7 \\ 6.8 \\ 6.9 \end{array}$	商用、先行事例など 1 自前でデータを用意する意味 1 地理情報の基礎知識 1 QGIS のインストール 1 データのダウンロード 1 データの読み込みと QGIS の基本操作 1 データの処理 その 2	71 73 73 77 78 85 90 92 102
$\begin{array}{c} 6.1 \\ 6.2 \\ 6.3 \\ 6.4 \\ 6.5 \\ 6.6 \\ 6.7 \\ 6.8 \\ 6.9 \\ 6.10 \end{array}$	間用、先行事例など 自前でデータを用意する意味 地理情報の基礎知識 QGIS のインストール データのダウンロード データの変換 データの読み込みと QGIS の基本操作 データの処理 その2 Unity から使う	73 73 73 77 78 85 90 92 102 115

著者



121

第1章

Cygwin 環境で Docker を快適に 使うために

1.1 はじめに

Cygwin は、Unix ライクな環境に慣れきっている人々が Windows を利用するにあたっ て救いとなるソフトウェアなのは皆さんご存知のところと思います^{*1}。また、モダンな開 発シーンでは Docker を使うことも増えてきました。しかし、Cygwin 環境から Docker を使おうとすると後述するいくつかの問題にぶつかります。

この章では、どうしても Cygwin から離れられない筆者のような人が、Cygwin 環境の まま Docker を快適に使えるようにする方法についてまとめます。また、試行錯誤の過程 を Qiita に投稿してありますので、合わせてご覧ください*²。

環境について

動作確認は Windows 7 上の Docker Toolbox で行っています。バージョンに制限は特に無いはずです。Windows 10 Pro と Docker for Windows の組み合わせでは確認をしていませんが、動作原理的には同じように使えると思われます。

1.2 Cygwin で Docker を使うときの問題点

Docker Toolbox にはデフォルトのターミナルとして Docker Quickstart Termnal (GitBash) が付属しています。しかし普段 Cygwin で生活しているのであれば、Cygwin の環境のまま docker コマンドを使いたくなってきます。

ドキュメント*³を参考に環境変数やパスの設定をすることで Docker Toolbox の docke

^{*&}lt;sup>1</sup> 現在では Cygwin 以外にも MSYS2 や Windows Subsystem for Linux など複数の選択肢があります。 よい時代になりましたね。

^{*2} https://qiita.com/makiuchi-d/items/7f177cc90c75af40e5ad

^{*3} https://docs.docker.com/machine/reference/env/

rコマンドを呼び出すことはできるようになります。ただ、このままでは次の2つの問題 点があり、実用できるとはとてもいえません。

TTY オプションが使えない

mintty (Cygwin の標準ターミナル)を使っている場合に docker runや docker exe cで疑似 TTY オプション (--ttyあるいは-t)を指定したとき、TTY モードを有効にで きないというエラーが発生します。

```
$ docker run -it ubuntu
cannot enable tty mode on non tty input
```

これは他のターミナルエミュレータを使うことで回避することもできますが、mintty のままで回避する方法を「1.3 TTY モードを有効にできない問題を回避する」で解説し ます。

そのままのパスをパラメータに指定できない

Cygwin 環境は独自のルートディレクトリを持ち、さらに Windows の各ドライブ を/cygdrive 以下に配置するという独特のディレクトリ構成になっています。しかし doc kerコマンドは Cygwin のアプリケーションでは無いため、Cygwin 環境のファイルパス をそのまま dockerのパラメータに指定しても、それは存在しないパスとして解釈されて しまいます。

この問題の回避方法について「1.4 Cygwin 環境のパスをそのまま指定できるようにする」で解説します。

1.3 TTY モードを有効にできない問題を回避する

この問題は、mintty のサイト*4にも書いてあるように、mintty の疑似 TTY が Windows のネイティブコンソールと互換を持たないためです。Windows のネイティブコンソール アプリケーションを mintty で動かすには、winpty*5を使って疑似 TTY とネイティブコ ンソールの橋渡しをする方法もありますが、ここでは筆者が最終的にたどり着いた sshコ マンドを利用する方法について解説します。

^{*4} https://mintty.github.io/#compatibility

^{*5} https://github.com/rprichard/winpty

SSH コマンドによるネイティブコンソールの回避

さて、Docker は Linux カーネルの機能に依存しているため、Windows 上で直接動作す ることはできません。Docker Toolbox や Docker for Windows では Linux の仮想マシン (VM)を動かし、その中で Docker が立ち上がっています。そして VM 内には dockerコ マンドも用意されており、VM にログインして dockerコマンドを叩くことができます。

この VM では sshdも立ち上がっているため、一般的な SSH クライアントで接続する ことができます。そして Cygwin の sshコマンドには-t(仮想 TTY)オプションがある ため、これで mintty の TTY と VM 内の dockerコマンドの TTY を直接つなぐことが できます。

つまり、sshコマンドで直接 VM 内の dockerコマンドを叩くことにより、Windows ネ イティブの docker.exeを使う必要がなくなり、ネイティブコンソールを完全に回避でき るのです*6。

ラッパースクリプトによる実装例

SSH を使って VM 内の docker コマンドを叩くものを、ラッパースクリプトとして実装してみます。このとき、docker runや docker execで疑似 TTY オプションが指定されている場合は、sshにも-tオプションをつけて呼び出すようにします。

接続する VM の IP アドレスやユーザ名、使用する鍵といった情報は、docker-ma chine inspectコマンドで取得することができます。ここに含まれるファイルパスは Windows の形式になっているため、Cygwin のパスに忘れずに変換しておきましょう。

▼リスト 1.1 SSH を使って docker コマンドを叩く bash スクリプト

```
#!/bin/bash -e
cmd=(docker "$0")
ttymode=false
# exec/run で -t/--tty があるときのみ ttymode=true
case "$1" in
    "exec" | "run")
    shift
    while [[ $# -gt 0 && $1 =~ ^- ]]; do
        if [[ "$1" =~ ^-[^-=]*t || "$1" =~ ^--tty(=true)? ]]; then
            ttymode=true
        fi
        if [[ "$1" =~ ^-[^-=]*t=false || "$1" == "--tty=false" ]]; then
            ttymode=false
        fi
        shift
        done
```

^{*6} Linux の仮想マシンが動いてさえいえれば、VirtualBox だろうと他の仮想化技術だろうと使える方法です。Docker Toolbox とはなんだったのか。

```
;;
esac

# SSH 接続に必要な情報を集める

INSPECT=$(docker-machine inspect)

SSHUSER=$(echo "$INSPECT" | grep -Po '"SSHUser":\s*"\K[^"]*(?=",)')

SSHKEY=$(echo "$INSPECT" | grep -Po '"SSHKeyPath":\s*"\K[^"]*(?=",)' |\

sed -E 's|^(\w+):|/cygdrive/\L\1|;s|\\\|/|g')

VMADDR=$(echo "$INSPECT" | grep -Po '"IPAddress":\s*"\K[^"]*(?=",)')

# ttymode 有効のときだけ ssh -t にする

sshopt=""

if $ttymode; then

sshopt="-t"

fi

ssh $sshopt -q -i "$SSHKEY" "$SSHUSER"@"$VMADDR" "$(printf "%q " "${cmd[@]}")"
```

このスクリプトを dockerという名前で PATH の通った場所においておけば、普段と まったく同じ感覚で dockerコマンドを利用できるようになります。winptyを使った場合 と違い、標準入出力をパイプにしても問題なく動きます。

もしかしたら docker-machineコマンドの応答待ち時間が気になるかもしれません。 複数の VM を切り替えるようなことが無いのであれば、接続先情報を固定値にしてしま うことでさらに快適にすることもできます。ぜひお試しください。

1.4 Cygwin 環境のパスをそのまま指定できるようにする

docker run時のボリュームのマウント (--volumeまたは-v) などで Windows 上の ファイルやディレクトリを dockerコマンドに渡すことはよくあります。VM 内で動作し ている Docker は、ホストである Windows のファイルやディレクトリにどうやってアク セスしているのでしょうか。

答えは単純で、共有フォルダとして VM 内にマウントしているのです。Docker Toolbox の場合、初期設定で Windows の C:\Users を VM 内の/c/Users にマウントしています。 ということは、C:\Users 以外の場所を渡すことは初期設定のままではできません。

一方、特にボリュームのマウントではフルパスでの指定を求められるのですが、これが VM 内でのフルパスになっていないと Docker は解釈できません。

Cygwin 環境でのフルパスは、C:\Users は/cygdrive/c/Users となってしまい、VM 内の フルパスとは異なります。加えて Cygwin の/home 以下などのパスについても、/c/Users 以下の対応するパスに書き換える必要があります。

ここでもまたラッパースクリプトで、dockerコマンドに渡す前にパスを変換すること で回避していきます。

Cygwin 環境のパスを VM 内のパスに変換する

Cygwin に付属している cygpathコマンドで、Cygwin 環境のパスをさまざまな形式の パスに変換することができます。特に cygpath -amとすると、ドライブレターから始ま る"/"区切りのフルパスが得られ、都合がよいです。



このようにシンボリックリンクも解決してくれるので、先頭のドライブレター部分を小 文字にして"/"で始まるように置換するだけで済みます。これを dmpathという関数とし て定義したものをリスト 1.2 に示します。

▼リスト 1.2 cygwin 環境のパスを VM 内のパスに変換するコマンド

```
function dmpath ()
{
    cygpath -am "$1" | sed -E 's|^(\w*):|/\L\1|'
}
```

パスが与えられるパラメータ部分だけパスを変換する

コマンドラインのパラメータを列挙しただけでは、どれが変換すべきパスなのか判別が つきません。それどころか、docker cpでは2つのパラメータのうち ":"の無い方だけ をパス変換しなければならない一方で、docker run -vの場合は ":"の前半のみパス変 換しなければならなず、さらに、コンテナ内で実行するコマンドのオプションについては 変換してはならないなど、対応方法が千差万別です。

正しく動作させるためには、知りうる限りのオプションを片っ端からチェックしていく 以外になさそうです。

ここでは docker cpと docker runの一部のオプションについてパス変換したものを リスト 1.3 に示します。

▼リスト 1.3 適切なパラメータのパスを変換して docker を呼び出す bash スクリプト

```
#!/bin/bash -e
# dmpath()の定義は省略
cmd=(docker "$@")
case "$1" in
```

```
"cp")
       cmd=(docker cp)
       shift
       while [[ $# -gt 0 ]]; do
           case "$1" in
               -* | *:*) # オプションやコンテナ内のパスは変換しない
                  cmd=("${cmd[@]}" "$1")
                   ;;
               *) # それ以外は変換する
                   cmd=("${cmd[@]}" "$(dmpath "$1")")
                   ::
           esac
           shift
       done
   ;;
"run")
       cmd=(docker run)
       shift
       while [[ $# -gt 0 && $1 =~ ^- ]]; do
           # volume オプションはホスト側のパス部分を取り出して置換
           if [[ "${1%=*}" =~ ^-[^-]*v$ || "${1%=*}" = "--volume" ]]; then
               case "$1" in
                  *=*)
                      opt="${1%=*}"
                      paths="${1#*=}"
                      ;;
                   *)
                      opt="$1"
                      paths="$2"
                      shift
                      ;;
               esac
               cmd=("${cmd[@]}" "$opt" "$(dmpath "${paths%:*}"):${paths#*:}")
           # 引数を取るオプションは次のパラメータも一緒に処理(一部)
           elif [[ "$1" =~ ^-[^-]*[aelpuw]$ ||
                  "$1" = "--attach" ||
                  "$1" = "--env" ||
                   "$1" = "--label" ||
                   "$1" = "--link" ||
                  "$1" = "--mount" ||
                   "$1" = "--name" ||
                  "$1" = "--publish" ||
                  "$1" = "--user" ||
                   "$1" = "--workdir" ]]; then
               cmd=("${cmd[@]}" "$1" "$2")
               shift
           # その他のパラメータはそのまま追加
           else
               cmd=("${cmd[@]}" "$1")
           fi
           shift
       done
       # パラメータ以外(image 名以降)は置換せず追加
       cmd=("${cmd[0]}" "$0")
       ;;
esac
$(printf "%q " "${cmd[@]}")
```

このスクリプトでは、docker runのパラメータを網羅できていません。たとえば、デ タッチキーを変更する--detach-keysは追加のパラメータを要求しますが、このままで は単独で使われるパラメータとして判定されてしまい、キー文字列をイメージ名と判定してしまいます。

少なくとも自身で使うパラメータはすべて網羅しなければなりません。もしエレガント な解決策などありましたらご教示ください。

また、ここに示した docker cp、docker run以外にもパスを受け取るコマンドがあり ます。それらについても同様の方法でパス変換することで、この問題は解決できます。

このようなラッパースクリプトを用いることで、パス名に関して Cygwin 環境であることを意識することなく dockerコマンドが使えるようになります。

1.5 まとめ

Cygwin 環境で dockerコマンドを使うときにぶつかる、疑似 TTY の問題とパス名の 問題について回避策を示しました。

紙面と解説の都合上、それぞれの回避策を別々のスクリプトとして紹介しましたが、この2つの回避策を統合し、さらにコマンドパラメータをより多く網羅したものを GitHub にて公開しています*7。

これにより Cygwin 環境でも快適に Docker を利用できるようになりました。お困りの 方はぜひご利用ください。

最後に強く言いたいことは、Docker を使うならホストも Linux にしましょう。

Makiuchi Daisuke / @makki_d

^{*7} https://github.com/makiuchi-d/docker-in-cygwin

第2章

Unity で開発しているスマホゲーム で Xcode のバージョンアップを する

KLab 入社から約三年が過ぎようとしているクライアントエンジニアです。KLab では これまで、Unity を使ったいくつかのプロジェクトでビルド環境、フローの改善に取り組 んできています。

本章では、これまで Xcode のバージョンアップを行ってきた経験を活かし、同じよう にビルド環境、フローの改善に取り組んでいる方々に向けた情報をお伝えしたいと思いま す。KLab で筆者が参加してきたプロジェクトの事例を通し、どのようなアプローチを取 るべきか? なるべく詳細に記載することで、読者の皆様の取り組みに役立てていただけ たらと思います。

本章を読むことで、次のような知見の獲得が期待できると思います。

- Unity で開発している iOS/Android アプリのビルドの知見
- iOS アプリのビルド手法、注意すべき点、対応ノウハウ
- Xcode バージョンアップ対応の TODO やスケジューリングの留意点

それでは、早速見ていきましょう。

2.1 Unity で開発しているスマホゲームのビルド手法

Unity で開発しているアプリのビルド手法

Unity でアプリのバイナリを生成する流れは大まかに図 2.1 のようになっています。



▲図 2.1 Unity で開発しているアプリのバイナリ生成フロー

Unity でビルドする際にはバッチビルドの手法が用いられます。バッチビルドとは、ビ ルドスクリプトを C#で記述して、コマンドラインで実行するものです。

最小限のビルドスクリプトの例をリスト 2.1 で紹介します。

▼リスト 2.1 最小限の Unity バッチビルドスクリプト

```
using UnityEditor;
class MyEditorScript
{
    static void PerformBuild ()
    {
       string[] scenes = { "Assets/MyScene.unity" };
       BuildPipeline.BuildPlayer(scenes, ...);
    }
}
```

このスクリプトをコマンドラインで実行する際はリスト 2.2のようにして実行します。

▼リスト 2.2 最小限の Unity バッチビルドスクリプト

```
/Applications/Unity/Unity.app/Contents/MacOS/Unity \
    -quit \
    -batchmode \
    -executeMethod MyEditorScript.PerformBuild
```

以上のスクリプトは Unity の公式ドキュメント*1 で紹介されている内容です。

実際のプロジェクトでは、前述の PerformBuild のパラメータに「どのサーバー環境に 接続するか」「In-App Purchase や Game Center のサンドボックス課金用バイナリか」

^{*1} Unity 公式ドキュメント

https://docs.unity3d.com/ja/current/Manual/CommandLineArguments.html

「本番バイナリか」といった内容の情報を含めているところが多いと思います。 参考までに、筆者が関わってきたタイトルで使っていたスクリプトを紹介します。

▼リスト 2.3 リズムゲームプロジェクトの Unity バッチビルドスクリプト







2.2 iOS アプリのビルド手法

前節では Unity でアプリのバイナリをバッチビルドで生成する流れを紹介しました。 本節では iOS アプリのビルドを具体的に見ていきましょう。

iOS アプリのビルドのしかたには大きくふたつあります。ひとつは「Xcode の GUI 画 面を用いたビルド(以降 GUI ビルドと呼びます)」です。そしてもうひとつは「Xcode Command Line Tool を用いたビルド(以降 CLI ビルドと呼びます)」です。

ipa の GUI ビルドの流れ

ipaのGUIビルドの流れは次のようになります(Xcode 9.2の場合)。

- 1. Xcode 上でプロジェクトを開く (図 2.3)
- Build Settings にて、ipa のビルドに使用するプロビジョニングプロファイル*2、 証明書を指定する(図 2.4)
- 3. Xcode > Run > Archive メニューを選択する (図 2.5)
- 4. Xcode > Organizer メニューを選択する
- 5. 画面に表示されたアプリの一覧から、archive を選択し、Export ボタンを選択する (図 2.6)
- アプリの配布方式の中から、2. で指定したプロビジョニングと合致するものを選 択し、Next ボタンを選択する(図 2.7)
- 7. App Thining の設定をして Next ボタンを選択する (図 2.8)
- 8. アプリのプロビジョニングプロファイル、証明書の中から、2. で指定した内容と合 致するものを選択し、Next ボタンを選択する (図 2.9)
- 9. 内容を確認して、Next ボタンを選択する (図 2.10)
- ipa や plist を export するディレクトリ階層や、ディレクトリ名を指定して Export ボタンを選択する(図 2.11)
- 11. ipa が生成されたことを確認する

^{*2} バイナリと開発端末、証明書を紐づけて管理するためのファイルで、拡張子は.provisioningprofile です。 エディタで開くことで端末リストや証明書の内容を確認することができます

H Q A O H D 0 Dete plobalgamemanagers plobalgamemanagers	E Ceneral Ceneral				< 4 >	D m
Deta globalgamemanagers globalgamemanagers, assets	Ceneral General				x - <i>x</i>	
Deta globalgamemanagers globalgamemanagers.assets		Capabilities Resource Tay	ps into Build Settings	Build Phases	Build Rules	Identity and Type
globalgamemanagers globalgamemanagers.assets	PROJECT					Name Unity-IPhone
globalgamemanagers.assets	D Linity different	* identify				Location Absolute
A set of the set of th	a contraction					
popergamemanagers assets ress	TARGETS	Display Name	Dev			Full Parts Aligensite/Lungage A/
level0	Unity-IPhone	Burnetin Interesting	and block			Documents/Project/
level1	Unity-IPhone Tests	and the second	concern.			
level2		Version				
level3		Build				Project Document
level4						And and a second s
level5						Project Format Accore 3.2 - Comparison
C Investig		* Signing				Organization
Contract of Contra			C a second of a second station			Class Prefix
- HORT			 Accomption of manage signing Young will make and under configure 	Constitution of the		
aven .			ouriFicates.			Text Bettings
aver9						Index Using Spaces
level10		an ann an that an an an an				many A.C.
level11		V Signing (Release)				Tab Ind
level12		Provisioning Profile	Dev	80		Wrap lines
level13			Contraction and the second			
level14		Team	KLab Global Pte. Ltd.			
level15		Signing Certificate	Phone Developer			
level16						
invel17		100000000000000000000000000000000000000				
local S.B.		* Signing (Debug)				
local 10		Provisioning Profile	Dev	0		
Charles and the second se						
			KORD GIODER PHP. CHL.			
Never 2 1		Signing Certificate	Whone Developer			
WVW22						
HV8/23		V. Destination to be to				
leve/24		· Depicyment and				
level25		Devloyment Ternet	70			
levs/26						
level27		Devices	Universal			000
level28						
level29						
level30						
level31		Main Interface				
level32		Device Orientation	Derival			
invel33		Dence of Brance	C Lincide Down			No Matches
invalid.			C Landscate Left			No Matches
level 15			Landscape Right			
1000						
Charles and Charle		Status Bar Style	Default	•		
analy i			C and do a state of the state			
84638			C PODE SCRUCK CHT			

▲図 2.3 Xcode 上でプロジェクトを開く

	100 d h				
DH420800	28 C 2 Long-smore	•		< a >	00
Unity-IPhone	D Oore	al Capabilities Resource Tags Info	Build Settings Build Phases Build Rules		Identity and Type
V Deta	PROJECT	Basic Customized All Company Leve	a + O- signing	0	Name Unity-IPhone
grobergemeinanagers	Linity-iPhone				Location Absolute
globelgememenegers assets	TABOUTS				
goosgamenanagers.assets.reso	Charles allowed	* Linking			Full Puth (Users/mizusewa-k/
level0	Georg-amona	Betting	Unity-Phone		Documents/Project/
level1	Unity-Phone Tests	Compatibility Version			
leve(2		Current Library Version			
wverd					Project Document
level4		* Signing			Project Format Xcode 3.2-compatible
level5		Betteg	Unity-Phone		Organization
levels		Code Signing Entitlements	(Users/mizusews-k/Documents/Project/	-	Class Prefix
level7		V Code Signing identity	iPhone Developer: 0		
leveld		Debug	iPhone Developer: 0		Text Bettings
level9		Any IOS SDK C	Phone Developer C		Indext Using Spaces
level10		Release	Phone Developer: 0		winner A T
level11		Any IOS SOR ()	interesting 2		Tab Indent
level12		Cold opping sign	Wigh Stabul Star Stat 1		Wrap lines
level13		Other Code Storing Flats			
level14		Provisioning Profile	Dev 0		
level15		Provisioning Profile (Deprecated)	Automatic 0		
level16					
level17		V Apple LLVM 9.0 - Warnings - All Janguages			
level18		Setting	Unity-Phone		
level19		Charlest Barrier Environments	No. 7		
level20		of a surface state state state			
level21					
level22		 Apple LLVM 8.0 - Warnings - Objective C and ARC 	and the second se		
level23		bring	may-more		
level24		Repeatedly using a _weak reference	NO C		
level25					
level26					
level27					D 0 0 B
level28					
level29					
level30					
level31					
level32					
level33					No Matches
level34					The material
level35					
level36					
level37					
level28					
					100 (in)

▲図 2.4 Xcode 上でプロビジョニングプロファイル、証明書を指定する

Run	ЖR
Test	HU
Profile	жı
Analyze	ΰжв
Archive	
Build For	•
Perform Action	•
Build	жв
Clean	<mark>ዮ</mark> ଞ
Stop	Ж.
Scheme	•
Destination	•
Create Bot	

▲図 2.5 Xcode 上で Archive メニューを実行する

		Arct	nives Crashes	
IOS Apps	Name	Creation Date	~ Version	Archive Information
	Lunity-iPhone	2018/03/26 11:59		Unity-iPhone 2018/03/26 11:59
				Upload to App Store
				Validate Export
				Details
				Version
				Identifier com.klab.
				Type IOS App Archive
	1			Download dSYMs
				Description
				No Description
	Tiltor	1a	rchive	

▲図 2.6 Xcode 上で ipa を作成する

Apps	Select a method of dist	ribution:		iformation
	3	App Store Distribute through the App Store. Ad Hoc Install on designated devices. Enterprise Distribute to your organization. Development Distribute to members of your ter	ım.	rie 159 vgp Store Export Archive dSYMs
	Cancel		Previous	Next

▲図 2.7 Xcode 上で ipa を作成する

S Apps	Development distribution options:		formation
			ne 1:59
			opp Store
		_	Export
	App Thinning: None Additional Options: Include manifest for over-the-air Users can download your app using Sa	installation _{ffari} .	-
			Archive
	3		dSYMs
	Cancel	Previous	xt cription
			_

▲図 2.8 Xcode 上で ipa を作成する

pps	Select certificate and iOS App Development profiles:		formation
			ne 1:59
			op Store
	Team: KLab Global Pte. Ltd.		Export
	Distribution certificate: [IOS Developer – Created 20	017/09/15	
	bleach.app: 🖉 Dev	0	Archive
			dSYMs
	3		
	Cancel	Previous Next	cription

▲図 2.9 Xcode 上で ipa を作成する

S Apps	Review Unity-iPhone.ipa c	ontent:		oformation
	.app	.app		ne 1:59
		SUMMARY Team: KLab Global Pte. Ltd.		App Store
		Certificate: IOS Development (Expires Profile: Dev ©	; 2018/09/15)	Export
		Architectures: armv7 and arm64		Anthin
		application-identifier .com.klab. aps-environment development		dSYMs
		com.apple.developer.associated-do applinks:bbs.onelink.me	mains	
	Cancel		Previous	ort cription

▲図 2.10 Xcode 上で ipa を作成する

		Export As:	Unity-iPhone 2018-03-26	12-12-24	<u>^</u>	
		Tags:				
			Documents	0		Q. Search
Recents Cloud Drive A Applications Desktop	Name				Size Kind	Date Added
Documents Documents Google ドライブ Provisioning Pr bin vices Remote Disc hared	Ī					
New Folder						Cancel Expor

▲図 2.11 Xcode 上で ipa を作成する

ipa の CLI ビルドの流れ

ipaの CLI ビルドの流れは次のようになります。

- 1. Xcode のプロジェクトをマシンに用意する
- 2. プロジェクトに対して、archive 生成コマンドを実行する
- 3. archive が生成されたら、archive に対して ipa エクスポートコマンドを実行する
- 4. ipa が生成されたことを確認する

Xcode のプロジェクトをマシンに用意する

Git からクローンする、Unity からプロジェクトを出力するなどして、Xcode プロジェ クトを用意します。

プロジェクトに対して、archive 生成コマンドを実行する

用意した Xcode プロジェクトに対して、リスト 2.4 のような archive 生成コマンドを 実行します。

▼リスト 2.4 Xcode CLI での archive ビルドスクリプト

xcodebuild \
-project "\${XCODE_PROJECT_CONFIG_PATH}" \
-configuration "\${CONFIGURATION}" \
-scheme Unity-iPhone \
-archivePath \$ARCHIVE_PATH \
-sdk iphoneos11.2 \
CODE_SIGN_IDENTITY="\${CODE_SIGN_IDENTITY}" \
PROVISIONING_PROFILE="\${PROFILE_UUID}" \
archive 2>&1)

archive に対して、ipa エクスポートコマンドを実行する

生成した archive に対して、ipa エクスポートコマンドを実行します。

▼リスト 2.5 Xcode CLI での ipa ビルドスクリプト

```
xcodebuild -exportArchive \
-archivePath $ARCHIVE_PATH -exportPath $UNITY_BUILD_PATH \
-exportOptionsPlist $EXPORT_OPTIONS_PLIST_PATH
```

ipa が生成されたことを確認する

生成された ipa を確認します。

2.3 Xcode バージョンアップのケーススタディ

本節では、バージョンアップする時の流れに沿って、どういう点に留意すべきか? 具体的に何を確認しながら進めるべきか? 示していきたいと思います。

Xcode バージョンアップの進め方(一例)

- 1. スケジュール作成
- 2. 検証用ビルドマシン、ジョブの確保
- 3. 検証用ビルドマシンの環境構築(Git・Unity・Xcode など)
- 4. ビルド環境検証(ローカルでの GUI/CLI ビルド・Jenkins ビルド)
- 5. 検証環境で作成したビルドのテスト
- 6. 運用ビルドマシン、ジョブへの反映

スケジュール作成

経験的に、おおよそ1週間から2週間程度見ておくと大体収まる印象です。前述した進 め方にのっとると、各項目の工数は次のようになります。

- 1. スケジュール作成: ---
- 2. 検証用ビルドマシン、ジョブの確保:0.5 日
- 3. 検証用ビルドマシンの環境構築 (Git・Unity・Xcode など):0.5 日
- 4. ビルド環境検証 (ローカルでの GUI/CLI ビルド・Jenkins ビルド):2.0 日
- 5. 検証環境で作成したビルドのテスト:1.5日
- 6. 運用ビルドマシン、ジョブへの反映:0.5日

工数を見る時に参考にしたいのは、一回のビルドにかかる時間です。Xcode をアップ デートした場合、ビルド手順や設定の確認と修正のために何度も試行錯誤する必要があり ます。しかし、プロジェクトごとにビルド時間は異なるとは思いますが、一度頭からビル ドするとなると全体で 30~60 分くらいかかります。場合によっては1日に8本程度しか ビルドを試せません。このため、比較的余裕を持ったスケジュールとしては、作業者が1 人の場合は2週間くらい見ておいた方が安心、という印象です。

検証用ビルドマシン、ジョブの確保

Xcode でのバイナリビルドについては、「異なるバージョンの Xcode で同時にビルドさ せようとすると片方のビルドが失敗してしまう」現象が知られています。このため、デイ リービルドなどを行なっているプロジェクトであれば、なるべく検証用のマシンを別途確 保したいところです。

また、Jenkins を利用しているプロジェクトの場合は、既存のジョブをコピーしてテス ト用のジョブを作成すると簡単に検証に取り掛かることができます。既存のジョブをその まま実行させながらジョブの設定を試行錯誤することができますので、テスト用ジョブを 確保しておくことはオススメです。

検証用ビルドマシンの環境構築

Xcode・Unity・Git など、必要なソフトをインストールします。このとき、macOS の バージョンアップも必要になる点は意識しておきたいポイントです。Xcode のメジャー バージョンアップではほとんどと言っていいほど macOS のバージョンアップが必要にな りますので、macOS の変更点の確認もしておいた方がよいでしょう^{*3}。

ビルド環境検証

GUI でのビルドは比較的すんなり行きやすいことと、CLI でのビルドに活かせる部分 があるので先に GUI でのビルドを試します。

^{*&}lt;sup>3</sup> Xcode を8系から9系にアップデートした際、「当時の最新 macOS マシンで、Unity がプロジェクト のファイルを認識できなくなる」現象が起きたのですが、その時は社内の空いているマシンの中から正常 動作する macOS バージョンのマシンを探して利用するという事例がありました

iOS ビルドでは、あるある話なのですが、大体コード署名エラーで失敗するのでその辺 りを見ていくことになります。このとき、エラーログの内容だけではわからないことが多 く、何だかんだで Xcode のリリースノート、デベロッパーフォーラムなどの力を借りる ことが多いです。また、プロビジョニングプロファイルや証明書、秘密鍵のインストール 方法によってビルドの成否が変わることもちょくちょくあります。

このときチェックしたいポイントを見ていきます。

プロビジョニングプロファイル

まず、プロビジョニングプロファイルの確認です。

Xcode が認識しているプロビジョニングプロファイル群は、/Users/{username}/Library/Mobile Device/ProvisioningProfile/以下にインストールされています。 ls /Users/{user-name}/Library/Mobile Device/ProvisioningProfile/というコマンドを ターミナルで実行すればプロビジョニングプロファイルの一覧が確認可能です。

no ProvisioningProfile といった「コード署名でプロビジョニングプロファイルが存在 しない」という旨のエラーが出てビルドに失敗する時は、ここにプロビジョニングプロ ファイルが入っているか確認しましょう。

また、まれに古いプロビジョニングプロファイルが残っていて悪さをしている場合もあ るので、一度一式削除して入れ直すのもよい手です。プロビジョニングプロファイルは、 次の方法でインストールすることができます。

- Xcode > Preference > Account に Apple Developer アカウントを設定し、Xcode で直接ダウンロード、インストールする
- Apple Developer Center*4にログインし、必要なプロビジョニングプロファイル をダウンロードし、ファイルをダブルクリックしてインストールする
- Apple Developer Center にログインし、必要なプロビジョニングプロファイルを CLI 上でインポートしてインストールする

^{*4} https://developer.apple.com/account/

Apple ID:	
Description:	@klab.com @klab.com
Team	Role
KLab Global Pte. Ltd. KLab INC. (ENT) KLab Inc.	Member Member Member
	Team KLab Global Pte. Ltd. KLab INC. (ENT) KLab Inc.

▲図 2.12 Xcode でのプロビジョニングプロファイルー括ダウンロード画面

証明書、秘密鍵

次に、証明書、秘密鍵の確認です。macOS が認識している証明書、秘密鍵はキーチェー ンアクセスで確認できます。

証明書で確認すべき点は次です。いずれかの状態に反している場合、署名に失敗する ケースがあります。

- コード署名証明書、中間認証局の証明書それぞれの有効期限が切れていないこと
 (図 2.13・図 2.14)
- Apple 中間認証局の証明書(Apple World Developer Relations Certificate Authority, Developer ID Certification Authority) がログインキーチェーンに登録されておらず、システムに表示されていること(Jenkins で CLI ビルドするマシンの場合のみ*5。図 2.15)

^{*5} GUI ビルドする場合は Apple 中間認証局の証明書がログインキーチェーンに登録されていても問題なく ビルドできます

•••	キーチェーンアクセス			
01909a200	ブインキーチェーンがロックされます。		Q (8.8)	
キーチェーン ログイン ローカル項目 システム マンテムルート	Crappent Phone Distribution: KLab Global Pte. Ltd 発行に、Apple Worldwide Developer Relations Certi 教授期後に、2017年11月24日金曜日17時30分279日 O この証明書の作業期間が知れています	. (FGAT3C6ZAC) fication Authority 本標單時		
	名前	へ 相助	変更日	有効期限
	iMessage Encryption Key	公開鍵		
	iMessage Encryption Key	公開館		
	IMessage Encryption Key	公開鍵		**
	iMessage Encryption Key	秘密鍵		
	IMessage Encryption Key	秘密鍵		**
	IMessage Encryption Key	秘密鍵		
分類	iMessage Signing Key	公開鍵		
S ##7058	P IMessage Signing Key	公開鍵		
A Provide	iMessage Signing Key	公開鍵		
	iMessage Signing Key	秘密鍵		**
物性メモ	iMessage Signing Key	秘密鍵		
自分の証明書	iMessage Signing Key	秘密篇		**
Y R	Phone Developer: Taro Noguchi (EX73GJX495)	証明書	**	2018/09
証明書	👘 🙀 iPhone Distribution: KLab Global Pte. Ltd. (FGAT3C6ZAC)	証明書	144	2017/11/
	iPhone Distribution: KLab INC. (ENT)	証明書		2020/12
	Ψ KG	秘密鍵		**
	¥ KLab Global Pte. Ltd.	秘密鍵		**
	. MetadataKeychain	アプリケーションパス…	2018/02/21 18:11:49	**
	4. Safari Session State Key	アプリケーションパス…	2018/01/25 18:11:08	
	+ 1 FF3	36008		

▲図 2.13 有効期限が切れた証明書をキーチェーンアクセスで確認している画面

• •	キーチェーンアクセス			
00001882D	ブインキーチェーンがロックされます。			Q, 19.11
キーチェーン ログイン ローカル頃目 システム システムルート	KLab Sysadm CA Gen.1 retain r			
	名前	48.00	有効期限	~ キーチェーン
	KLab Sysadm CA Gen.1	复利者	2030/01/01 8:59:59	ログイン
	Developer ID Certification Authority	証明書	2027/02/02 7:12:15	ログイン
	Apple Application Integration Certification Authority	証明書	2026/10/21 9:00:00	ログイン
	13	298	2025/10/19 20:11:26	ログイン
		証明書	2025/02/10 9:18:14	ログイン
		証明書	2024/11/15 14:09:24	ログイン
98	Apple Worldwide Developer Relations Certification Authority	証明書	2023/02/08 6:48:47	ログイン
1012030	Apple Phone Certification Authority	証明書	2022/04/13 2:43:28	ログイン
a coordina		証明書	2019/06/29 8:18:04	ログイン
K. バスワード	> D	臣明書	2019/05/18 11:18:21	ログイン
秘密メモ	IPhone Distribution: KLab Global Pte. Ltd. (FGAT3C6ZAC)	証明書	2018/10/12 19:17:33	ログイン
自分の証明書 賃	iPhone Developer: Taro Noguchi (EX73GJX495)	証明書	2018/09/15 14:34:02	ログイン
証明書				
	+ i ace 1	2.後日		

▲図 2.14 ログインキーチェーンに登録されている証明書を確認している画面

	キーチェーンアクセス			
000088252	テムキーチェーンがロックされます。			Q, 19.11
キーチェーン ログイン ローカル項目 システム	Apple Worldwide Developer Relations Certification Authority essezza essez essezez essezezezezezezezezezezezezezezezeze			
C PATAN-P	6.0	~ 800	有於原題	キーチェーン
	KLab Personal CA Gen 1	12 10 10	2030/01/01 8 59 59	0294
		臣明書	2035/02/10 19:54:28	9274
	10	正明書	2035/02/10 19:54:29	227A
	Apple Worldwide Developer Relations Certification Authority	ENB	2023/02/08 6:48:47	2376
分類 数 ずべての項目 ん パスワード - 税表メモ - 自分の証明書 - 鍵 - 2 証明書				

▲図 2.15 システムキーチェーンに登録されている証明書を確認している画面

プロビジョニングプロファイル同様、古い証明書がまれに悪さをしている場合もあるので、一度一式削除して入れ直すのもよい手です。証明書は、次の方法でインストールすることができます*6。

- Apple Developer Center にログインし、必要な証明書をダウンロードし、ファイ ルをダブルクリックしてインストールする
- Apple Developer Center にログインし、必要な証明書を CLI 上でインポートして インストールする (リスト 2.6)

秘密鍵で確認すべき点は次です。

 秘密鍵の 情報を見る > アクセス制御 項目で、アクセスすることを許可したア プリケーションとして/usr/bin/codesign が追加されていること(図 2.16)

^{*&}lt;sup>6</sup> 一般的に、iOS アプリの証明書は秘密鍵を含む p12 ファイルで扱います

-	KG	
	属性アクセス制御	
この項目の使用をすべ	てのアプリケーションに許可	
🧿 アクセスを許可する前	に確認	
= キーチェーンパスワ	ワードを要求	
これらのアプリケーション	によるアクセスを常に許可:	
名前		
codesign		

▲図 2.16 証明書に紐づけられた秘密鍵の情報確認画面で codesign にアクセスを許可しているか確認している画面

▼リスト 2.6 証明書を CLI 上でインポートする

```
# キーチェーンのアンロック
security unlock-keychain -p $OSX_ADMIN_PASSWORD "${KEYCHAIN_LOCATION}"
security set-keychain-settings -t 3600 -1 "${KEYCHAIN_LOCATION}"
# インポート
security import "${IOS_P12_FILE_PATH}" \
    -f pkcs12 \
    -P "${IOS_P12_PASSWORD}" \
    -k "${KEYCHAIN_LOCATION}" \
    -T /usr/bin/codesign
```

iOS アプリ開発に必要になる証明書

一般的に、iOS のコード署名証明書は、「証明書」の情報に「CSR^aを生成した PC の秘密鍵」の情報が付加された p12 ファイルで扱うことが多いと思いますが、この p12 ファイルは以下のような手順で作ることができます。

- 1. Mac のキーチェーンアクセスで certSigningRequest ファイルを生成
- で生成した certSigningRequest ファイルを Apple Developer Center に アップロードして cer ファイルを生成
- 3. 2. で生成した cer ファイルを 1. の作業を行った PC (certSigningRequest を生成した PC) でダウンロード
- 4. 3. でダウンロードした cer ファイルを実行して、キーチェーンアクセスに 登録
- 5. 4. で登録した証明書をキーチェーンアクセスで選択して、p12 ファイルと して書き出し生成

手順こそ少々複雑ですが、このように p12 ファイルの形で扱うことによって certSigningRequest ファイルを生成した PC 以外でも証明書を利用できるように なり、その結果ビルドマシンごとに証明書を生成する必要がなくなるメリットが あります。

^a Certificate Signing Request。証明書作成を認証機関に依頼するために作成が必要になるファ イルです

2.4 Xcode バージョンアップによるビルド手法の変更点、 キャッチアップ方法

ここまで、駆け足ですが Unity で開発している iOS アプリのビルド手法、バージョン アップ時の注意点などを示してきました。

Xcode のバージョンアップでは、それぞれ微妙にビルド方法に変更が入り、ビルドスク リプトなどに手を入れる必要が生じます。

大きくは中間成果物 (app・xcarchive など) を作成して、署名して ipa を生成する流れ ですが、コマンドの必須オプションが増えたり、オプションで指定するファイルを別途用 意する必要が出て来たりとバージョンにより違いがあります。

筆者の今までの経験的には、たとえば「app を生成して ipa を作る方法から xcarchive を生成して ipa を作る方法に変更した (Xcode $6.3 \rightarrow 7.1$)」「ipa 生成時のオプションで署 名の種別などを定義する plist (ExportOptions.plist) を指定するように変更した (Xcode

7.1 \rightarrow 8.2)」「ExportOptions.plist の必須定義を追記した、ストア用のアプリアイコンを バイナリに追加した (Xcode 8.2 \rightarrow 9.2)」といったものがあります。

これらは Xcode のリリースノートなどのドキュメントにも記述されているものですが、 普通にビルドを試して出てきたエラーログを眺めるだけでは分かりにくいものです。公式 ドキュメントは読んでおきましょう。

Xcode のバージョンアップが必要になる状況とは

そもそも、Xcode のバージョンアップは何のために行うのでしょうか。 Xcode のバージョンアップが必要になる状況というのはプロジェクトによってま ちまちです。基本的には新しい API を使えるようにする、App Store で継続的に アプリをリリースし続けられるようにする、ということだと思いますが、昨今の スマフォゲームでは多くの外部 SDK を利用していますので、この SDK の動作状 況に引っ張られてやむなくアップデート、ということもあります。 以下は筆者が携わったプロジェクトのバージョンアップ経緯です。

- Xcode 6.3 → 7.1:パズルプロジェクトの場合
 - パズルプロジェクトでは、新しい iOS のベータ版で、広告 SDK の初期化でアプリが落ちる現象が起きていたため Xcode バージョンアップ を行う必要がありました。
 - SDK のバージョンアップでの対応が期間的にできず、また Xcode の バージョンアップにより現象が発生しないことがわかっていたため、 Xcode バージョンアップを行うことは必須でした。
- Xcode 7.1 → 8.2:リズムアクションゲームプロジェクトの場合
 - リズムアクションゲームプロジェクトの場合、新 iOS の API 利用の 需要が強くあったため、新 iOS の SDK が同梱されているより新しい Xcode にバージョンアップを行うことが望ましい状況でした。
- Xcode $8.2 \rightarrow 9.2$: アクションゲームプロジェクトの場合
 - iPhone X でのフルスクリーン表示のため。
 - レターボックス表示の場合、将来的にヒューマンインターフェイスガ イドラインに沿っていないことによりアプリの審査に落ちてしまうこ とが想定されたため。

2.5 おわりに

Xcode のバージョンアップは、6 系から7 系、7 系から8 系、そして8 系から9 系と三度に渡り経験してきました。何だかんだでいまだにつまづく時はつまづくし、ハマって時間を費やす時は費やします。

iOS ではアプリの審査があり、動作確認用だけでなく iTunes Connect にアップロード できるバイナリも作る必要もありますので、さらに時間がかかります。また、アプリが利 用しているサードパーティの SDK についても、動作に問題がないかなどのテストをして おくことも必要です。Xcode のアップデートは必要になってから対応を進めるよりは、一 年に一度のメジャーアップデートのタイミングで早めに済ませておくのもひとつの手かも しれません。

最後になりますが、実際のところ、こうした記録としてまとめておくことで自身の Xcode バージョンアップ作業の備忘録、助けになることを期待したりもします。しかしな がら、本章の内容が、自分自身だけではなく読者の方々の役に立つことができれば嬉しい なと思います。

Kinuko MIZUSAWA



第3章

Re:VIEW で書いた原稿の CI を AWS CodeBuild で回す

KLab でビルド、Jenkins 周りも見ているクライアントエンジニアです。この章では、 この本(本書「KLab Tech Book Vol. 2」)の pdf 生成 CI 環境を作ったよ。という話を したいと思います。

3.1 ことのはじまり

発起人岡本から、「技術書典3のときはできなかったけど今度こそ CI 環境ちゃんと作りたい! 誰かやりたい人いない??」という声かけがあったので、ちょうど「クラウド CI サービス、そういえばちょっと興味あるな〜」と思いやってみることになりました。

今回は、CI 環境を支えるサービスとして AWS CodeBuild^{*1}を使っています。当初は Travis CI を利用する話も挙がったのですが、社内では AWS を積極的に使っているので、 社内実績のない CodeBuild も使ってみれると良さそうということや、Travis CI よりも費 用が安くできるのではということがあり、検証して問題なければ採用してみるという流れ でそのまま採用となりました。

3.2 CI環境、フロー

CI フロー図

CIのフローは図 3.1のようになります。

^{*1} AWS CodeBuild https://aws.amazon.com/jp/



▲図 3.1 AWS, GitHub Enterprise を利用した Re:VIEW 原稿の CI フロー

- GitHub Enterprise の Webhook でコミットがプッシュされるたびに pdf 生成
- 生成された pdf は Slack で通知

システム構成図

システムの構成は図 3.2 のようになります。



▲図 3.2 AWS, GitHub Enterprise を利用した Re:VIEW 原稿の CI システム構成

- GitHub Enterprise $^{\ast 2}$
 - Re:VIEW の原稿の置き場所
- AWS CodeBuild
 - 原稿の pdf 化を行う場所
- AWS S3*3
 - 証明書、成果物の置き場所
- AWS Elastric Container Service $(ECS)^{*4}$
 - pdf 化を行うマシンの Docker*5イメージの置き場所
- Slack^{*6}
 - 技術書典 pdf のアップロード先

AWS では、CodeBuild で作られた成果物は同じ AWS の S3 に保存することができ、 ECS に Docker イメージを置いておくことで、独自の Docker イメージで CodeBuild の ビルド処理を走らせることもできます。2018 年 1 月末より GitHub Enterprise もサポー トされ、Webhook によるビルドができるようになっていたため、今回はそれらを組み合 わせた形で構成しています。

^{*2} GitHub Enterprise https://enterprise.github.com/home

 $^{^{*3}}$ AWS S3 https://aws.amazon.com/jp/

^{*4} AWS Elastric Container Service https://aws.amazon.com/jp/

^{*5} Docker https://www.docker.com/

^{*6} Slack https://slack.com/
3.3 AWS CodeBuild とは



▲図 3.3 AWS CodeBuild トップ画面

Amazon が提供している AWS の中に含まれている CI サービスです。

特徴

ビルドにかかった時間によって請求金額が決まり、ビルドしなければ費用が発生しない 点が大きな特徴です。また、AWSの他サービスとの連携がサポートされていて扱いやす い点も特徴といえます。また、日本語のドキュメント、チュートリアルも比較的充実して いる印象です。

料金体系、用意されているマシンスペック

利用するコンピュータのスペックによって、分単位で決まります。ビルド時間は分ごと に切り上げで扱われます。

インスタンスタイプ	メモリ	CPU 数	1 分あたりの料金 (USD)
build.general1.small	3 GB	2	\$ 0.005
build.general1.medium	$7 \mathrm{GB}$	4	\$ 0.010
build.general1.large	$15~\mathrm{GB}$	8	\$ 0.020

また、プログラミング言語環境としては、予め必要なツールがインストールされた Docker イメージとして、言語ごとに Ubuntu 14.04 系で用意されています。CodeBuild では AWS の他サービスを利用することで自前のカスタム Docker イメージを利用するこ とが可能ですが、特に必要がなければまず、CodeBuild 標準の Docker イメージを利用で きないか検討するとよさそうです。

CodeBuild では、以下の言語をインストールした Docker イメージをバージョンごとに 個別に用意しており、言語環境を選ぶ形で Docker イメージを選択することができます。

- Android Java 8 24.4.1
- Docker 17.09.0
- Go 1.7.3
- Java 8
- Node.js 6.3.1, 4.4.7, 4.3.2
- Python 3.5.2, 3.4.5, 3.3.6, 2.7.12
- Ruby 2.3.1, 2.2.5
- .NET Core 1.1, 2.0

3.4 GHE / AWS (S3, CodeBuild, ECS) を利用した Re:VIEW CI 環境構築

GHE / AWS (S3, CodeBuild, ECS) を利用した Re:VIEW CI 環境構築 の流れ

CI 環境構築の手順はざっくり次のようになります。

- 1. Re:VIEW プロジェクトを追加(GHE)
- 2. 証明書・成果物保管用ストレージを AWS 上に作成(S3)
- 3. CI の実行環境となる Docker イメージを AWS 上に構築 (ECS)
- 4. CI プロジェクトを作成 (CodeBuild)
- 5. CI ビルドスクリプトを作成
- 6. CI を試行
- 7. リポジトリの Webhook 設定
- 8. 流れの確認

手順 1 Re:VIEW プロジェクトを追加(GHE)

Re:VIEW で init して作っておきます。ルートに buildspec.yml というファイル名で、 ビルド定義ファイルを追加しておきます。このビルド定義ファイルの内容は、Codebuild のプロジェクト作成画面のコピーで作成可能です。参考までに、リスト 3.1 に buildspec.yml の初期内容を紹介します。

```
▼リスト 3.1 参考: buildspec.yml の初期内容
```

```
version: 0.2
#env:
  #variables:
    # key: "value"
     # key: "value"
  #parameter-store:
     # key: "value"
     # key: "value"
phases:
  #install:
    #commands:
      # - command
      # - command
  #pre_build:
    #commands:
      # - command
      # - command
  build:
    commands:
      # - command
      # - command
  #post_build:
    #commands:
      # - command
      # - command
#artifacts:
  #files:
    # - location
    # - location
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
   # - paths
}
```

手順 2 証明書・成果物保管用ストレージを AWS 上に作成(S3)

今回は、GitHub Enterprise の Webhook を利用するため、CodeBuild のビルドを実行 するイメージに証明書をインストールできるようにする必要があります。CodeBuild で は、この証明書を S3 のバケットと呼ばれるストレージに置き、CodeBuild 側で参照する ように設定することで、証明書のインストールを CodeBuild のプロセスの中で自動的に 行ってくれます*7。

ここではS3バケットの作成フロー、手順について紹介します。

S3 バケットの作成フロー

■S3 のトップ画面を開く AWS トップ画面から S3 のリンクを選択し、S3 のトップ画 面を開きます。

■バケットを作成する S3 のトップ画面で「バケットを作成する」を選択すると、バケットの作成をすることができます。バケットの作成では「バケット名」「リージョン名」を 入力する必要があるため、順番に入れていきます。

aws +-ビス - リソースク	ループ 、 🖌 🗘	mizusawa-k 🖗 🛛 👻 👻	グローバル ・ サポート ・
Amazon S3 へようこそ。新しいパケ	バケットの作成		× F#1X>>
🎼 Amazon S3	 (1) 名前とリージョン ② プロパティの設定 ③ アパま 	クセス許可の設 👍 硫酸	ご存知でしたか?
Q パケット検索 + パケットを作成する パケットを	名前とリージョン パケット名 🕕		
パケット名 十三	mizusawa-k-test		t£.
8	リージョン		2014 8:25:42 午後 - 1900
8	アジアパシフィック (東京)	×.	2013 5:34:53 午後 900
8	 既存のパケットから設定をコピー		2013 5:34:43 午後 900
8	パケットを選択する (省略可)	21 パケット 🗸	2, 2013 10:30:28 MT+0900
8			2017 5:56:05 午後 900
8), 2013 8:29:15 午 [+0900
), 2011 2:29:24 午 [+0900
			2013 4:37:53 午後 900
8			3, 2017 12:54:29 MT+0900
8	作成	キャンセル 次), 2011 5:59:11 午 1+0900
8	182492	京)	年前 GMT+0900

▲図 3.4 AWS S3 バケット作成画面

「バケット名」は、今回は mizusawa-k-test としました。 「リージョン」は、今回は**アジアパシフィック(東京)**としました。 設定を進めていき、確認して問題なければ「バケットを作成」を選択します。

^{*7} 今回利用した用途以外にも、S3 はビルドの成果物となる原稿の pdf データをクラウドサーバー上に保管 しておきたい場合にも活用できます

Amazon S3 へようこそ。新しいパケ		バケットの作成		× ^{F‡} ±×>
Mazon S3	✓ 名前とリージョン	プロパティの設定 🕜 アクセス 定	許可の設 ④ 確認	ご存知でしたか?
Q パケット検索	名前とリージョン		# #	
+ パケットを作成する パケットを	パケット名 mizusawa-k-te	et リージョン アジアパシフィッ		- <i>3</i> 37 3
パケット名 †三	プロパティ			1 <u>=</u> -
8	バージョニング	無効		2014 8:25:42 午後 1900
8	サーバーアクセスのログ記	■ ■ 無効 ● なが		2013 5:34:53 午9 900
	ッッ オブジェクトレペルのログ 録	099 配 _{無効}		2013 5:34:43 午餐 900
8	デフォルト暗号化	なし		2, 2013 10:30:28 MT+0900
	アクセス権限		編集	2017 5:56:05 午後 900
8	ユーザー パプリックアクセス許可	2 無効), 2013 8:29:15 午 F+0900
8	システムのアクセス許可	fm(3t)), 2011 2:29:24 午 [+0900
				2013 4:37:53 午餐 900
8				3, 2017 12:54:29 MT+0900
8			戻る パケットを作用	-), 2011 5:59:11 午 2 「+0900
				8, 2017 12:45:48

▲図 3.5 AWS S3 バケット作成確認画面

手順 3 CI の実行環境となる Docker イメージを AWS 上に構築 (ECS)

Re:VIEW をビルドする Docker イメージは、vvakame さんが提供している物*⁸を利用 させてもらいました。

ECS リポジトリの作成フロー

まずは ECS のリポジトリを作成します。ECS のリポジトリ作成には、次のコマンドラ インツールが必要なので、インストールしておきましょう。

- AWS CLI*9
- Docker
- pip, python

インストールしたら、順に進めていきます。

^{*8} Re:VIEW image for Docker https://hub.docker.com/r/vvakame/review/

^{*9} AWS CLI https://aws.amazon.com/jp/cli/

■ECS のトップ画面を開く トップ画面を開きます。

■リポジトリの作成を選択する リポジトリの作成を選択します。

■リポジトリの設定 リポジトリの設定をして作成します。項目は「リポジトリ名」ひと つだけです。

ステップ 1: リポジトリの設定	リポジトリの設定	
ステップ 2: Docker イメージの構築、タグ付け、プッシュ	このウィザードでは、Elastic Container Registry にリポジトリを作成する手順が示されます。詳細は	256
	リポジトリ名* gijutsushoten 0	
	名前空間はオプションであり、スラッシュを使用してリボジトり名に含め ができます(namespace/repo など)。	525
	リポジトリの URI	
	northeast- 1.amazonaws.com/gijutsushoten	
	アクセス許可	
	所有者には、デフォルトでこのリポジトリへのアクセスが許可されます。このウィザードの完了後に ソールでこのリポジトリに対するその他のアクセス権限を付与できます。	. =>

▲図 3.6 AWS ECS リポジトリ作成画面

「リポジトリ名」には、識別できる名前を入れておきます。

■次へを選択する 次のステップを選択すると、AWS CLI のコマンドが表示されます。 また、コマンドと合わせて、Docker イメージのプッシュ方法のヘルプページへのリンク も表示されています。リンク先のページは「Amazon ECR の使用開始*¹⁰」というヘルプ ページです。

^{*10} Amazon ECR の使用開始

https://docs.aws.amazon.com/ja_jp/AmazonECR/latest/userguide/ECR_GetStarted.html

aws サービス v リソースグループ v 🖈	♪ mizusawa-k @ v 東京 v サポート v
Elastic Container Registry の開	始方法
Elastic Container Registry の開 ステップ1: リポジトリの設定 ステップ 2: Docker イメージの機築、ダグ付け、フッシ ュ	始方法 Docker イメージの構築、タグ付け、プッシュ JTボントリが作成されたところで、以下の手属で Docker イメージをブッシュできます: UTジーリが作成されたところで、以下の手属で Docker イメージをブッシュできます: UTジーリングになった。 UTジーリングになった。 UTジーリングになった。 UTジーリングになった。 UTジーリングを取得します。このコマンドは、レジストリに対して Docker クライアントを認 こまるために使用できます: UTVDACF Togin コマンドを実行します。 Dimのステップで送された docker login コマンドを実行します。 Dimのなのテップで送された docker login コマンドを実行します。 Dimのステップで送された docker login コマンドを実行します。 Dimのステップで送された docker login コマンドを使用している場合は、この手属を省略できます: Dimotheast-1) Dimotheast-1 Dimotheast
	5)以下のコマンドを実行して、新しく作成したAWSリボジトリにこのイメージをブッシュします: docker push
	*ðā <u>27</u>

▲図 3.7 AWS ECS リポジトリ作成コマンド確認画面

■コマンドを実行していく コマンドを順番に、ターミナルで実行し、リポジトリにイ メージをプッシュしていきます。ひととおりコマンド作業が終わると、AWS ECS のコン ソール画面でイメージが確認できるようになります。

■CodeBuild で利用できるようにアクセス許可を設定する 初期状態だと CodeBuild で 利用できないアクセス状態になっているので、許可するように設定を追加していきます。

ECS のリポジトリを作成すると、AWS CLI で実行するプッシュコマンドが表示され るので、コマンドの実行を進めます。ちなみに、macOS で Docker イメージを作成する 場合は VM の起動が必要なので、要注意です。

手順 4 CI プロジェクトを作成(CodeBuild)

CodeBuild の設定はプロジェクトという単位で行います。プロジェクトの設定を順番 に進めていきます。

aws サービス - リソースグル-	フ~ *	🗘 mizusawa-k 🛛	▼ 東京 マ サポート マ
プロジェクトの作成			
ステップ 1: プロジェクトの設定	プロジェクトの設定		0
ステップ 2: 確認	ビルドプロジェクト用の設定を指定します	r.	
	プロジェクト名*	プロジェクト名の入力	0
	88.49	 説明の追加 	
	ソース: ビルドの対象		
	ソースプロバイダ・	ソースプロバイダーの選択	
	環境: ビルド方法		
	環境イメージ・	 AWS CodeBuild によって管理された イメージの使用 Docker イメージの指定 	
	オペレーティングシステム*	オペレーティングシステムの選択	
	特撒付与	Docker イメージを構築するか、ビル ドで昇格されたアクセス権限を取得す るには、このフラグを有効にします。	r
	ビルド仕様	 ソースコードのルートディレクトリの buildspec.yml を使用 ビルドコマンドの挿入 	2
	buildspec 名	buildspec.yml	0
	証明書	自己署名証明書または証明機関によって 署名された証明書がある場合は、S3パケ ットからその証明書をインストールする オプションを選択します。 E期書をインストールしない	
	アーティファクト: このビルドブ	ロジェクトからアーティファクト	を配置する場所
	タイプ・	プロジェクトのアーティファク	0
	キャッシュ		
	タイプ・	キャッシュなし	
	サービスロール		
	Ann'S CodeBuild かお各様に优美する AWS	マカウントでサービュロールを作が	へロールを加定します。 評問はこちち.
	ō	アカウントから既存のサービスロールを選	织
	ロール名*	サービスロール名の入力	
	VPC		
	VPC*	AWS CodeBuild プロジェクトからアクセ スする VPC を選択します。詳細情報 No VPC	0
	 詳細設定の表示 		
	•必須		キャンセル 装行
マイードバック 〇 日本語	© 2008 - 2018,	Amazon Web Services, Inc. or its atfiliates. All rights	i reserved. プライパシーポリシー 利用規約

▲図 3.8 AWS CodeBuild プロジェクト作成画面

プロジェクトの設定

■プロジェクト名 識別しやすい名前を入れておきます。今回は、mizusawa-k-test と しました。

■説明 プロジェクトの用途、目的などを入れておきます。今回は、検証用プロジェクト ということを明記しておきました。

ソース: ビルドの対象

■ソースプロバイダ 次のバージョン管理システムの中から、どのシステムでプロジェク トファイルを取得するか選択することができます。

- Amazon S3
- AWS CodeCommit
- Bitbucket
- GitHub
- GitHub Enterprise

今回は、GitHub Enterprise を選択しました。

環境: ビルド方法

■環境イメージ 次のイメージを選択することが出来ます。

- AWS CodeBuild によって管理されたイメージの使用
- Docker イメージの指定

今回は、すでに用意した Docker イメージを利用するため、**Docker イメージの指定**を 選択しました。

■環境タイプ Linux のみ選択可能なので、Linux を選択しました。

■カスタムイメージタイプ 次のイメージタイプを選択することができます。

- Amazon ECR
- その他

今回はすでに作成してあるリポジトリを選択するため、Amazon ECR を選択します。

■Amazon ECR リポジトリ すでに作成してあるリポジトリを選択します。今回は gijutsushoten-4 を選択します。

■特権付与 インストールなどで必要な場合があるので、チェックを入れておきます。

■現在のビルド仕様 ソースコードのルートディレクトリに buildspec.yml を置くので、 そのままにしておきます。

■証明書、証明書のバケット、証明書のオブジェクトキー 「証明書」項目にて、証明書 が必要かどうかで次の選択肢を選びます。

- 証明書をインストールしない
- S3 から証明書をインストールする

今回は Webhook で使用するため、S3 から証明書をインストールするを選択します。 「証明書のバケット」項目では、S3 のバケットを選択します。今回は mizusawa-k-test を選択しました。「証明書のオブジェクトキー」項目では、S3 のバケットにアップロード しておいた GHE の SSL 証明書である pem ファイルの名前を拡張子付きで入力します。 今回は、github-enterprise.pem と入力しました。

アーティファクト: このビルドプロジェクトからアーティファクトを配置する場所

■**タイプ** AWS CodeBuild では、ビルドの成果物をアーティファクトという名前で扱っ ています。「タイプ」項目ではこのアーティファクトをS3 に保存するかどうかを選択する ことができます。

- Amazon S3
- アーティファクトなし

今回は、ビルドの度に pdf を Slack に送信することから、S3 での保存が必要なかった ため、**アーティファクトなし**を選択しました。

キャッシュ

■**タイプ** AWS CodeBuild では、キャッシュの仕組みも持っているようです。次の選択 肢からキャッシュの保存場所を選択できます。

- Amazon S3
- キャッシュなし

今回は、S3 をあまり使わないようにする為、キャッシュなしを選択しました。

VPC

■VPC 特に使用しない為、No VPC を選択します。

詳細設定

その他、細かい設定が可能です。

■タイムアウト デフォルト(1時間)のまま

■暗号化キー デフォルトのまま

■コンピューティングタイプ

- 3 GB メモリ、2vCPU
- 7 GB メモリ、4vCPU
- 15 GB メモリ、8vCPU

15 GB メモリ、8vCPU を選択

■環境変数 特に設定なし

■タグ 特に設定なし

確認、作成

以上をまとめると、今回は図 3.9 のように設定しました。

ステップ 1: プロジェクトの設定	確認	6
ステップ 2: 確認	このビルドプロジェクトのパラメーターと	設定を確認します。
	プロジェクト	
	プロジェクト名 mi	zusawa-k-test
	說明 mi	zusawa-k の検証用プロジェクトです
	ソース	
	ソースプロバイダ Gi	Hub Enterprise
	リポジトリ	/gijutsushoten-4.git
	Git のクローンの深さ 1	
	Webhook tru	e
	バッジ 無	80
	Insecure SSL fail	50
	ビルド環境	
	環境タイプ Lir	ux
	イメージ	.dkr.ecr.ap-northeast-1.amazonews.com/gijutsushoten-4:1
	特権付な tru	e
	ビルド仕様 ピ	ルド仕様の表示
	証明書 mi	zusawa-k-test/github-enterprise.pem
	アーティファクト	
	917 7	ーディファクトなし
	キャッシュ	
	TT772	<i>π</i> ~ ² / − <i>t</i> 21.
	717 T	
	サービスロール	
	サービスロール名 an	naws:iam: role/service-role/codebuild-mizusawa-k-test
	詳細設定	
	ビルドタイムアウト 60	9
	暗号化キー デ	フォルトキー
	コンピューティングタイプ 30	A X T U SUCOU

▲図 3.9 AWS CodeBuild プロジェクト作成画面

確認して、問題なければ保存を押すとプロジェクトが作成されます。

手順5CIビルドスクリプトを作成

今回 AWS 上に構築した Docker マシンでは、必要なツールはすでに Docker イメー ジに入れてあります。また、原稿データを保管しているリポジトリのクローンは AWS CodeBuild 側で、ビルドスクリプトを実行する前にすでに行ってくれています。この為、 ビルドスクリプトの中で書かなければならないことは次のみになります。

- pdf 化
- pdf を Slack ヘアップロード

結果として、次のようなビルドスクリプトリスト 3.2 になりました。それぞれ、順番に 説明していきます。

▼リスト 3.2 参考: buildspec.yml の最終内容

```
version: 0.2
#env:
  #variables:
    # key: "value"
     # key: "value"
  #parameter-store:
     # key: "value"
     # key: "value"
phases:
  #install:
    #commands:
     # - command
  #pre_build:
    #commands:
      # - command
  build:
    commands:
      # build
      - . ./export-pdf.sh
  post_build:
    commands:
      # upload Slack
      - . ./upload-pdf-to-slack.sh
#artifacts:
  #files:
    # - locations
  #discard-paths: yes
  #base-directory: location
#cache:
  #paths:
    # - paths
```

pdf 化

Re:VIEW で.re ファイルなどをまとめて、pdf 化するコマンドを記述します。今回 は、.sh ファイルにコマンドをまとめておきました。

▼リスト 3.3 参考:pdf 化するシェルスクリプト

npm run pdf

pdf を Slack ヘアップロード

Slack への pdf アップロードには Slack の files.upload API^{*11} を利用します。この API の利用には token が必要ですが、token の生成は Slack のサイト内でできます^{*12}。

岸 slack Api		Documentation	Tutorials Your Apps
Home Building Slack apps Internal integrations Recent updates Best practices App blueprints	Legacy tokens You're viewing documentation on I into their Slack team. To securely ut the Events API, build internal integr started.	egacy custom integrations, an older way ilize the newest platform features like n ations as part of a Slack app just for you	for teams to build nessage buttons & ir team instead. Get
Custom integrations Legacy overview Incoming Webhooks Slash Commands	Legacy token generat	OT e tokens for testing and development.	
Bot Users Outgoing Webhooks Web API Legacy tokens	Legacy tokens are just for you applications. Do not publish Lessafety tips.	Never share legacy tokens with other users tokens in public code repositories.	isers or Review token
cogacy concis	By creating a test API token, you agree	e to the Slack API Terms of Service.	
	KLab Group mizusawa-k	xoxp-	Re-issue token

▲図 3.10 Slack Token 取得・再生成画面

今回は、pdf 化コマンドと同様、.sh ファイルにコマンドをまとめておきました。

^{*11} Slack files.upload API https://api.slack.com/methods/files.upload

 $^{^{*12} \ {\}tt Slack \ Legacy \ Tokens \ https://api.slack.com/custom-integrations/legacy-tokens}$

▼リスト 3.4 参考: pdf を Slack にアップロードするシェルスクリプト

```
# ディレクトリ移動
cd articles
# 版情報作成
HISTORY=$(grep -C 0 history: config.yml)
HISTORY=$(echo "$HISTORY" | sed 's/^.*"\(.*\)".*$/\1/' #)
echo ${HISTORY}
# Slack 投稿用コメント作成
SLACK_COMMENT="【"${HISTORY}"】 の原稿が出来ました"
# Slack 投稿コメント確認
echo "Slack 投稿コメント確認"
echo ${SLACK_COMMENT}
# Slack 投稿
curl -F file=@gijutsushoten-4-klab.pdf \
     -F channels=gr-kgtechblog \
     -F token=xoxp---- \
     -F initial_comment=${SLACK_COMMENT} \
     https://slack.com/api/files.upload
```

実際にポストしているのは curl のコマンドです。curl コマンドのパラメータにある to kenは、前述した Slack のサイトで生成した文字列を入れることになります。fileで指定 しているファイル名に@をつける必要がある点に注意です。channelsで指定するチャン ネル名には#をつける必要はないようです。

今回は、Slack 投稿時にファイルごとに版情報と Git のコミット ID が分かるように、i nitial_commentに含めるようにしておきました。

手順 6 CI を試行

ここまで来たら、一度ビルドを試行してみます。

ビルドの実行はボタンでできます。成功すると、SUCCESS と表示されます。途中、エ ラーが起きると即時にビルドが中止されて FAILED と表示されます。

	minungun le testes define	0 4-6- 40-4 00-	- E-14017E-	107-	
AWS CodeBuild	mizusawa-k-test:cedctbc	9-dabc-42c4-900	a-5042175e	9/C Succeeded	
ビルドプロジェク	処理の進行に応じてビルドの詳細を確認します。				
F	- #SBC(7				
ビルド履歴	1.2.0.12				
	ENF				
	ピルド ARN	amaws:codebuild:ap-northeast-	1: :build/m	izusawa-k-test.cedcfbc9-da6c-42c4-9	
		oca-oo42175ee97c			
		Oithèide Enternaine			
	URBEN	(Are de	tio b, ostode et éclide		
	ローンの運き	1	an gernen anen a ge		
	NUMBER OF CONTRACTOR	Mar 16, 2018 7:33:59 AM UTC			
	转了時間	Mar 16, 2018 7:35:06 AM UTC			
	ステータス	Succeeded			
	1=21-9	mizusawa-k			
	・ビルドの詳細				
	ノェース評細				
	68	ステータス	所要時間	完了済み	
	SUBMITTED	Succeeded		Mar 16, 2018 7:33:59 AM UTC	
	 PROVISIONING 	Succeeded	31 89	Mar 16, 2018 7:34:31 AM UTC	
	DOWNLOAD_SOURCE	Succeeded	29 89	Mar 16, 2018 7:35:01 AM UTC	
	 INSTALL 	Succeeded		Mar 16, 2018 7:35:01 AM UTC	
	PRE_BUILD	Succeeded		Mar 16, 2018 7:35:01 AM UTC	
	BUILD	Succeeded	1.89	Mar 16, 2018 7:35:02 AM UTC	
	POST_BUILD	Succeeded	189	Mar 16, 2018 7:35:04 AM UTC	
	UPLOAD_ARTIFACTS	Succeeded		Mar 16, 2018 7:35:04 AM UTC	
	FINALIZING	Succeeded	2.89	Mar 16, 2018 7:35:06 AM UTC	
	 COMPLETED 	Succeeded			
	ビルドログ				
	以下のビードログの最後の 10000 日をままゆ、ロ	ダ会体の書景			

▲図 3.11 CodeBuild で成功したビルド画面

aws サービス v	リソースグループ 🗸 🕏			mizusawa-k Ø	• #R • 7#-> •	
AWS CodeBuild	mizusawa-k-test:8c21211	5-c1fe-4519-ac	3b-8115439	681b3 mm	6	
ビルドプロジェク	処理の進行に応じてビルドの詳細を確認します。					
トビルド履歴	BMB					
	ビルド					
	ビルド ARN	am aws codebuild ap-northea c3b-8115439681b3	st-1: build	/mizusawa-k-test/8c212115-c1fe-4519-a		
	ビルドプロジェクト mizusawa-k-test					
	ソースプロバイダ	GitHub Enterprise /doujinshi/ggi.tsushoten-4.git				
	リポジトリ					
	Git のクローンの深さ	1				
	開始時間	Mar 11, 2018 7:15:50 AM UTC	2			
	終了時間	Mar 11, 2018 7:38:55 AM UTC				
	ステータス	Failed				
	イニシエータ	mizusawa-k				
	ビルドの詳細					
	フェーズ詳細					
	名前	ステータス	所要時間	完了読み		
	 SUBMITTED 	Succeeded		Mar 11, 2018 7:15:50 AM UTC		
	PROVISIONING	Succeeded	23 89	Mar 11, 2018 7:16:14 AM UTC		
	DOWNLOAD_SOURCE	Succeeded	16 89	Mar 11, 2018 7:16:31 AM UTC		
	 INSTALL 	Failed	22 33, 21 89	Mar 11, 2016 7:36:52 AM UTC		
	FINALIZING	Succeeded	2.69	Mar 11, 2018 7:38:55 AM UTC		
	COMPLETED	Succeeded				
	ビルドログ					
	以下のビルドログの最後の 10000 行を表示中。 ログ全体の表示					
	M9 54610 © 0 246310 0 0.02218 0.02210 0.02010 BRAK 99 34610 M9 34610 0 0.06614 0 0.02218 0.022108 0.00011 30818 M9 34810 M9 348300 0 0 0 0.022108 0.022111 0.00109 30948 M9 348300 0 0 0 0.22020 0.022210 0.02214 0.00010 30948 0 0 280500 0.022120 0.02214 0.00010 30948 0 48108 M9 34708 0 0 24630 0 0.022119 0.022117 0 0.000130	1804k 98 3481M 98 3443M 0 0 1222.04 0:00:13 1988k 99 348 6405k 0 0:22:20 0:22:09 0:0 4536M 0 0 2639k 0 0:22:20 0: 3481M 99 3470M 0 0 2661k 0 00:02 3565k 99 3481M 99 348	0 2662k 0 0122118 0122 81M 99 3645M 0 0 2663k 911 1983k 99 3683M 99 132112 0180108 1975k 9 0122112 0180108 1975k 9 0122119 0122115 01001 80M 0 0 2663k 0 012211	cd: 0:00116 18215 NB 34518 NB 34680 0 20423 0 0.022110 0:02207 0:00121 2055 NB 34801 NB 34 34584 0 0 26538 0 0:02210 0:02110 0:0610 1956 9 34828 NB 34638 0 0:02210 0:02110 0:0010 1055 9 34828 NB 345488 NB 34788 0 0:02210 0:02113 0 0 31148 NB 34681 NB 34788 0 0 26618 0 0:02218 8 0:022118 37884100 34818 100 34818 0	0:22:19 0:22:05 0:00:14 51M 0 0 2661x 0 0:22:15 x 99 3401M 98 3457M 0 0 2653 00:07 2104x 99 3481M 99 3465 0:22:16 0:00:03 3340x 99 2663x 0 0:22:18 0:22:18	

▲図 3.12 CodeBuild で失敗したビルド画面

手順7 リポジトリの Webhook 設定

SSL 証明書設定

SSL 証明書(.pem ファイル)を、ブラウザで GitHub Enterprise のリポジトリサイト にアクセスして取得します。取得するために必要な操作は、OS, ブラウザによって異なる ので注意が必要です。

000 mizusawa-k ×		θ
← ⇒ C ● 保護された通信	/mizusawa-k	☆ 🛄 🖉 😳 🗄
C Enterprise Search GitHub	GeoTrust Primary Certification Authority - 03.	+• ••
	RfT元: RapidSSL SH4256 CA - G2 有気期間: 2019年6月3日日間日 日本原準時	ture helps 🧨 Edit profile 🗙
	 ● この証明書は有効です ▶ 詳細な情報 	ing o
	Popular repositories	Customize your pinned repositories

▲図 3.13 ブラウザで GitHub Enterprise 画面を開き SSL 証明書をダウンロード



取得後は、作成した S3 のバケットにアップロードします。

▲図 3.14 SSL 証明書を S3 バケットにアップロード

a	WS サービス ~ リソースク	ガループ ~ 14	🗘 mizusawa-k Ø	- 70-	パル・サポート・
	Amazon S3 > mizusawa-k-test	;	アップロード		
-04	板要	 ファイルの面积 アクセス 定する 	許可を設 (3) プロパティを設定 する	(d) #18	
	キ アップロード + フォルダの作	1ファイル サイズ: 5.0 KB ターゲット ル	ス: mizusawa-k-test		(東京) 2
		+ さらにファイルを追加する			
	このハ	github-enterprise.crt		×	
	オブジェクトのアッ				アクセ
	パケットは、Amazon S3 に保存さ べてを収納するグローバルに一意 デナです。			×	のアクセ アクセス て他のユ ることが
	1748	IVE	N	IVE	
		97.CB	<i>06</i>		

▲図 3.15 SSL 証明書を S3 バケットにアップロード

aws サービス · リソースグ	リレーブ 🗸 🖌		🗘 mizusawa-k 👰	- /0	-パル - サポート -
Amazon S3 > mizusawa-k-test		アップロ	ч -к		
极要		アクセス許可を設定する	マロバティを設定 する	(4) #18	
土 アップロード + フォルダの作	ファイル				?(東京) 😂
	1ファイル	サイズ:5	.0 KB		
このハ	アクセス権限			-	
	1 被付与者				
	プロパティ			# \$	
	戦号化 いいえ メタデータ	ストレージク スタンダード	72		
オゴジークトのアッド	**				704
-F					200
パケットは、Amazon S3 に保存さ べてを収納するグローバルに一意 テナです。			R	る アップロード	、のアクセ アクセス 、て他のユ 「ることが
THE		IF HE		IT IE	
		今ずぐ始める			

▲図 3.16 SSL 証明書を S3 バケットにアップロード

アクセストークン生成

CodeBuild と GitHub のプッシュアクションを紐づけるために、GitHub のプロフィー ル画面、アクセストークンを生成します。

Personal settings	Public profile	
Profile	Name	Profile picture
Account		
Emails	Public email	10 March 10
Notifications	Select an email to display \$	
SSH and GPG keys	You can manage email addresses in your email settings.	
Security	Bio	
Repositories	Tell us a little bit about yourself	
Organizations		Upload new picture
Saved replies	You can @mention other users and organizations to link to them.	
Applications	URL	

▲図 3.17 GitHub アクセストークン生成: Personal setting > Profile 画面で Developer setting を選択

Enterprise Search Git	iub Pull requests Issues Explore	+• 🛄•
Settings / Developer settings		
OAuth Apps	Personal access tokens	Generate new token
GitHub Apps	Need an API token for scripts or testing? Generate a personal access toke	en for quick access to the GitHub API.
Personal access tokens	Personal access tokens function like ordinary OAuth access tokens. They can be use or can be used to authenticate to the API over Basic Authentication.	ed instead of a password for Git over HTTPS,

▲図 3.18 GitHub アクセストークン生成: Developer settings > Personal access tokens 画 面で Generate new token を選択

Settings / Developer settings		
OAuth Apps	New personal acce	ess token
GitHub Apps		
Personal access tokens	Personal access tokens function Git over HTTPS, or can be used	on like ordinary OAuth access tokens. They can be used instead of a password f d to authenticate to the API over Basic Authentication.
	Token description	
	ghe-webhook-build	
	What's this token for?	
	Select scopes	
	Scopes define the access for p	personal tokens. Read more about OAuth scopes.
	🕑 repo	Full control of private repositories
	repo:status	Access commit status
	repo_deployment	Access deployment status
	repo:invite	Access repository invitations
	admin:org	Full control of orgs and teams
	write:org	Read and write org and team membership
	read:org	Read org and team membership
	admin:public_key	Full control of user public keys
	write:public_key	Write user public keys
	read:public_key	Read user public keys
	admin:repo_hook	Full control of repository hooks
	vrite:repo_hook	Write repository hooks
		near repository induktion
	admin:org_nook	Full control of organization nooks
] gist	Create gists
	notifications	Access notifications
	🗆 user	Update all user data
	read:user	Read all user profile data
	user:email	Access user email addresses (read-only) Follow and unfollow users
	delete repo	Delete repositories
		Pentral enformament of are reactive locals for an organization
	admin:pre_receive_hook	control emorement of pre-receive nooks for an organization of repository
	admin:gpg_key	Full control of user gpg keys (Developer Preview)
	write:gpg_key	Title was you to fa

▲図 3.19 GitHub アクセストークン生成: Developer settings > Personal access tokens 画 面で repo にチェックを入れて Generate token を選択

Settings / Developer settings				
OAuth Apps	Personal access token	s	Generate new token	Revoke all
GitHub Apps	Tokens you have generated tha	at can be used to access the GitHub A	PI.	
Personal access tokens	Make sure to copy your new	personal access token now. You won'	't be able to see it again!	
			1	

▲図 3.20 GitHub アクセストークン生成: Developer settings > Personal access tokens 画 面で生成された token を確認

リポジトリの Webhook 有効化設定

リポジトリ自体に、Webhook の有効化設定を行います。GitHub のリポジトリの Webhook 設定画面を開き、CodeBuild で表示された「ペイロードの URL」と「シーク レット」を入力して設定します。

	Insecure SSL false	
	ビルド環境	
	ウェブフックの作成	×
	GitHub Enterprise でリポジトリのウェブフックを作成してください。 ペイロードの URL:	
	https://codebuild.ap-northeast-1.amazonaws.com/webhooks?	
	シークレット:	
	Concernance of the second	
		間じる
	キャンセル	戻る 保存してビルド 保存
フィードバック Q 日本語	© 2008 - 2018, Amazon Web Services, Inc. or	its affiliates. All rights reserved. プライパシーポリシー 利用規

▲図 3.21 GitHub Webhook 設定画面



When merging pull requests, you can allow any combination of merge commits, squashing, or rebasing. At least one option must be enabled.

▲図 3.22 GitHub Webhook 設定画面

🛛 doujinshi / gijutsushot	en-4
O Code 🕕 Issues (4)	Pull requests 0 Projects 0 B Wiki in Insights 🗘 Settings
Options	Webhooks Add webho
Collaborators & teams	Wehhnoks allow external services to be notified when certain events hannen. When the specified events hann
	regelegels another set they to be notified threat with stering hereing a section in the
Branches	we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.
Branches Hooks	we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide. We will also send events from this repository to your organization webhooks.
Branches Hooks Integrations & services	we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide. We will also send events from this repository to your organization webhooks. Pre-receive hooks
Branches Hooks Integrations & services Deploy keys	we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide. We will also send events from this repository to your organization webhooks. Pre-receive hooks
Branches Hooks Integrations & services Deploy keys Custom tabs	we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide. We will also send events from this repository to your organization webhooks. Pre-receive hooks Pre-receive hooks are scripts that run on the GitHub Enterprise server to enforce policy. When a push occurs, each script runs in an isolated environment to determine whether the push is accepted or rejected.
Branches Hooks Integrations & services Deploy keys Custom tabs	we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide. We will also send events from this repository to your organization webhooks. Pre-receive hooks Pre-receive hooks are scripts that run on the GitHub Enterprise server to enforce policy. When a push occurs, each script runs in an isolated environment to determine whether the push is accepted or rejected. No pre-receive hooks yet.

▲図 3.23 GitHub Webhook 設定画面

doujinshi / gijutsushote	O Unwatch - 178 ★ Unstar 1 ♀ Fork 0
O Code ① Issues 4	🗅 Pull requests 🗴 📃 Projects 🗴 📰 Wild 🔄 Insights 🕏 Settings
Options	Webhooks / Add webhook
Collaborators & teams	We'll send a POST request to the URL below with details of any subscribed events. You can also specify which
Branches	data format you'd like to receive (JSON, x-www-form-unlencoded, etc). More information can be found in our developer documentation.
Hooks	Pauload UPL *
Integrations & services	https://example.com/postreceive
Deploy keys	
Custom tabs	application/x-www-form-urlencoded \$
	Sacrat
	Which events would you like to trigger this webbook?
	Just the push event.
	Send me everything.
	C Let me select individual events.
	2 Active

▲図 3.24 GitHub Webhook 設定画面

手順8流れの確認

ここまで来たらひととおりの作業は完了です。コミットをプッシュして、CodeBuild プロジェクトのビルドが実行され、Slack に pdf が共有されるまでの流れを通してみま しょう。

3.5 CI 環境構築でハマったところ

ここまで CI 環境構築を進めていく中で、個人的にこれはハマったなというところをま とめておきます。

GitHub Enterprise の証明書のエクスポートができるブラウザが限定されていた

GitHub Enterprise の証明書をエクスポートで少々手こずりました。Mac マシンで作 業をしていたところ、ブラウザで証明書をエクスポートする際は、Safari, Chrome ではエ クスポートする UI が見当たらず、できませんでした。最終的には Firefox でエクスポー トして対応しました。

AWS CodeBuild プロジェクトの作成でアカウントの権限追加が必要 だった

プロジェクトの作成時、サービスロールを追加する必要があり、IAM アカウントの管 理者権限が必要だったので付与するように対応しました。

GitHub Enterprise のソース URL が https じゃないと失敗した

SSH の URL 指定だと、DOWNLOAD_SOURCE のフェーズで cannot run ssh エ ラーと出てビルドが失敗していたので、https の URL を指定して暫定対応しました。公 式ドキュメントでは例示の URL が https にはなっていましたが、特に注意書きがなかっ たためつまづいてしまいました。

AWS CodeBuild で用意している Docker (Ubuntu) に、日本語を含め た TeX のインストールができなかった

「パッケージ管理ソフトでインストールする」「ミラーサイトからダウンロードして展 開、インストールする」いくつかの方法を試したものの、日本語フォントなどを含めた パッケージのインストールができない、そもそもパッケージ検索で出てこない問題があり ました。また、提供されている Docker の Ubuntu のバージョンも 14.04 のみとなってい たことから、TeX 環境を自力で構築することは難しいと思いカスタムの Docker イメージ による環境構築で対応しました。

Slack のアップロードを Slack App(Bot) から行うことができなかった

当初は自分自身の Slack ユーザーアカウントではなく、Slack App (Bot) からアップ ロードしてもらう形で進めていたのですが、時間の都合上できませんでした。Bot で投稿 する際の注意点としては、Bot 用の Slack アプリを作成して、ファイルのアップロード権 限をつけ、Bot 用の Slack App の token で API を叩くようにする点だと思いますが、私 が API 実行を試した時には token エラーが返って来ていました。token の取得場所を間 違っていたか、どこかで見落としがあったのかなと思います。

今回はとりあえず pdf を Slack のチャンネルにアップロードできればいいかなというこ とで、ユーザーアカウントからのアップロードのままにしましたが、この場合にちょっと 怖いのは、アップロードしたユーザーアカウント自身が、自分がファイルをアップロード している事に気付きにくいところです。



▲図 3.25 Slack ファイルアップロードおよび通常の発言ログの表示画面

Slack ではチャンネルに未読の発言が投稿された際、チャンネル名がハイライト表示さ れるのですが、今回のようにファイルアップロードを API で実行した際にはチャンネル 名のハイライト表示はされません。API 実行時に指定している token 自体も、ファイル アップロード以外にも通常の発言にも使えるようですので、やろうと思えば、同じリポジ トリを見ている人がいたずらでなりすますこともできてしまいます^{*13}。Slack 画面の見か け上では、本人が発言したのか API で発言したのかわからないので、Slack のユーザーア カウントの token は Git のリポジトリに保管しない方が安全そうです。

3.6 おわりに

意外と細かいところでつまづいてしまい、真顔になったケースも多かったですが、なん とかみんなの CI 環境が整ってよかったです。AWS CodeBuild はビルド時間で費用が決 まる従量課金制なので、環境を作ってもビルドしなければそのまま放置しても費用が発生 しないという点は環境構築のコストを考えると地味に嬉しいところなんじゃないかなと思 います。

本章の内容が、技術書執筆、CI に関わる方々の御役に立てば幸いです。

Kinuko MIZUSAWA

^{*13} token の更新をすればひとまずなりすまし投稿を止めることはできます:)

第4章

WebAssembly で行列の演算を する

4.1 はじめに

皆さんは WebAssembly にどんな夢を抱いていますか? C/C++ 資産の活用、難読化、 高速化といった話を私はよく耳にします。

C/C++資産の活用は特にわかりやすい用途です。たとえば、ゲームエンジン の Unity^{*1}の WebGL 出力は IL2CPP^{*2}で出力された C/C++ ソースコードを Emscripten^{*3}を使用して WebAssembly に変換しています。特にゲームではディスプレイの 更新頻度を維持するため 1 フレームの処理をを 16ms 以内に終えることが求められます が、WebAssembly を使用すればそれを目指すこともできます。

一方でWebAssemblyを難読化目的に使うのはどうでしょうか。JavaScript はそもそ もとして解析しやすく、モダンなブラウザには優秀な JavaScript デバッグツールが搭載 されています。ゲームを実装するにあたって、ルールが解析されたり変数が書き換えられ ることは非常に大きな問題なのですが、JavaScript ではそれが簡単にできてしまいます。 そのため、JavaScript ゲーム開発者には解析されることや改ざんされることを想定した設 計が要求されます。WebAssembly はその点バイナリフォーマットなので JavaScript よ りは解析されにくいようにも思えます。ですが、WebAssembly のデコンパイルツールが 公式で公開されており、バイナリフォーマットから C 言語のソースコードへと簡単に戻 せてしまいます。バイナリだから安全とはいえません。

では、高速化という点はどうでしょうか。WebAssembly を使ったとしても、すべ ての処理が WebAssembly バイナリの中で行われるわけではありません。たとえば、 Emscripten を使用すれば OpenGL を使用したソースコードもコンパイルできます。し かし、Emscripten において OpenGL の API の大部分は JavaScript にて実装されていま

^{*1} https://unity3d.com/

^{*2} https://docs.unity3d.com/Manual/IL2CPP.html

^{*3} https://github.com/kripken/emscripten

す*⁴。つまりは、WebAssembly を使用していたとしても JavaScript の API を呼ぶこと を避けられません。

実際に WebAssembly を使用すると JavaScript より高速に処理できるのでしょうか、 ちょっと試してみたいと思います。

4.2 4行4列行列の乗算

ゲームグラフィクスでは4行4列の行列演算を多用します。

/	m_{00}	m_{01}	m_{02}	m_{03}	
	m_{10}	m_{11}	m_{12}	m_{13}	
	m_{20}	m_{21}	m_{22}	m_{23}	
/	m_{30}	m_{31}	m_{32}	m_{33})

ベクトルを回転させたり、移動させたりといった処理はこの行列演算を用いれば簡単 に行えます。回転する行列と移動する行列を乗算すれば回転した後に移動する行列を作 成できます。GPUにおいてもこの4行4列行列は使用可能で、大量の行列演算には大き な需要があります。この行列の乗算をを JavaScript と Emscripten で実装し、両者のパ フォーマンスを比較します。

4.3 JavaScript による実装

行列は WebGL でも使用する長さが 16 の Float32Arrayを使用します。これに列単位 に行列要素を配置していきます。先の行列の場合だと配列には [m00, m10, m20, m30, ..., m03, m13, m23, m33]のように配置します。

次の行列であればリスト 4.1 のように表現します。ちょうどソースコードと実際の行列 が転置の関係になっています。

$$\left(\begin{array}{rrrrr}1&2&3&4\\5&6&7&8\\1&2&3&4\\5&6&7&8\end{array}\right)$$

▼リスト 4.1 Float32Array による 4 × 4 行列

```
const mat1 = new Float32Array([
   1, 5, 1, 5,
   2, 6, 2, 6,
   3, 7, 3, 7,
   4, 8, 4, 8
]);
```

^{*4} https://github.com/kripken/emscripten/blob/master/src/library_gl.js

次に乗算を実装します。行列の乗算の実装は行と列をそれぞれ乗算したものを行列の成 分とします。これを JavaScript で実装するとリスト 4.2 となります。

▼リスト 4.2 JavaScript による 4 行 4 列行列の積

```
const mat1 = new Float32Array([
 1, 5, 1, 5,
 2, 6, 2, 6,
 3, 7, 3, 7,
 4, 8, 4, 8
]);
const mat2 = new Float32Array([
 1, 5, 1, 5,
 2, 6, 2, 6,
 3, 7, 3, 7,
 4, 8, 4, 8
1):
const matResult = new Float32Array([
 1, 0, 0, 0,
 0, 1, 0, 0,
 0, 0, 1, 0,
  0, 0, 0, 1
]);
MultiplyMatrixJS(mat1, mat2, matResult);
// [34, 82, 34, 82, 44, 108, 44, 108, 54, 134, 54, 134, 64, 160, 64, 160]
console.log(matResult);
```

4.4 Emscripten による実装

まずは Emscripten をインストールしてください*5。

Windows 10 で動作確認しましたが、Windows、Mac、Linux どの OS でも構いません。 Emscripten で JavaScript から呼び出す関数を定義する場合、マングリングされてしま うと呼び出しが難しくなるため、extern "C"で関数をマングリングしないようにします。

行列は JavaScript での実装と同様に列単位で要素を格納します。バッファは WebAssembly のヒープから取得するため、C++ 側でmallocを呼び出します。それぞれの要素を 16 個の引数に取るようにしました。

また、C++ 側で確保したバッファは C++ 側で解放する必要があります。これらをま とめてリスト 4.3 に示します。

 $^{^{*5}}$ https://kripken.github.io/emscripten-site/docs/getting_started/downloads.html

```
▼リスト 4.3 行列の作成と解放
```

```
// 行列の作成
extern "C" float *CreateMatrix(
    float m00, float m10, float m20, float m30,
    float m01, float m11, float m21, float m31, float m02, float m12, float m22, float m32,
    float m03, float m13, float m23, float m33)
ſ
  auto p = static_cast<float *>(malloc(sizeof(float) * 16));
  p[0] = m00; p[1] = m10; p[2] = m20; p[3] = m30;
  p[4] = m01; p[5] = m11; p[6] = m21; p[7] = m31;
  p[8] = m02; p[9] = m12; p[10] = m22; p[11] = m32;
  p[12] = m03; p[13] = m13; p[14] = m23; p[15] = m33;
 return p;
}
// 行列の解放
extern "C" void ReleaseMatrix(float *p)
Ł
  free(p);
3
```

行列の乗算も C++ で実装します (リスト 4.4)。左辺行列 a と右辺行列 b を受け取っ て作成済みの行列 dst に値をセットします。この実装はリスト 4.2 とほぼ同じ実装となっ ています。引数は float 型へのポインタとなっています。

▼リスト 4.4 行列の乗算

```
extern "C" void MultiplyMatrix(float *a, float *b, float *dst)
{
  for (auto x = 0; x < 4; x++)
  {
    for (auto y = 0; y < 4; y++)
    {
      auto i = x * 4;
      dst[i + y] =
            a[0 + y] * b[i + 0] +
            a[4 + y] * b[i + 1] +
            a[8 + y] * b[i + 2] +
            a[12 + y] * b[i + 3];
    }
}</pre>
```

これら WebAssembly の関数を呼び出す JavaScript で、Module オブジェクトを定義 します。WebAssembly の実行後に呼び出される postRun関数にて WebAssembly の関 数を取得します。JavaScript から WebAssembly の関数を呼び出すための関数を cwrap 関数で取得します。cwrap関数は関数名、リターン型、引数型を引数にとります。

▼リスト 4.5 JavaScript からのモジュール呼び出し

CreateMatrix関数は引数に行列要素を number型で取り、作成した行列をポインタを number型で返します。MultiplyMatrix関数の引数には CreateMatrixで取得したポイ ンタを渡します。このポインタは Module.bufferからのバイトオフセットとなっている ため、Float32Arrayを通じて配列アクセスができます (リスト 4.6)。

▼リスト 4.6 CreateMatrix から Float32Array を取得

```
const pMat1 = CreateMatrix(
   1, 5, 1, 5,
   2, 6, 2, 6,
   3, 7, 3, 7,
   4, 8, 4, 8
);
// [1, 5, 1, 5, 2, 6, 2, 6, 3, 7, 3, 7, 4, 8, 4, 8]
const mat1 = new Float32Array(Module.buffer, pMat1, 16);
```

ビルド時はリスト 4.7 のコマンドでビルドします。

EXPORTED_FUNCTIONSにエクスポートする関数を列挙するのですが、この際に関数名の 先頭にアンダースコアをつけるのを忘れないようにします。また、JavaScript からランタ イムの cwrap関数を呼び出すために EXTRA_EXPORTED_RUNTIME_METHODSに cwrapを列 挙します。

▼リスト 4.7 Emscripten でのビルド

```
em++ m4mul.cpp -o m4mul.js \
   -Wall \
   -std=c++11 \
   -03 \
   -g0 \
   -s WASM=1 \
   -s EXPORTED_FUNCTIONS="[ \
    '_CreateMatrix', \
    '_MultiplyMatrix', \
    '_ReleaseMatrix' \
]" \
   -s EXTRA_EXPORTED_RUNTIME_METHODS="[ \
    'cwrap' \
]"
```

ビルドした WebAssembly を使用して行列の乗算を行ってみました(リスト 4.8)。リ スト 4.2 と同じ結果を得ることができました。

▼リスト 4.8 Emscripten での演算結果

```
const pMat1 = CreateMatrix(
 1, 5, 1, 5,
 2, 6, 2, 6,
 3, 7, 3, 7,
 4, 8, 4, 8
);
const pMat2 = CreateMatrix(
 1, 5, 1, 5,
 2, 6, 2, 6,
 3, 7, 3, 7,
 4, 8, 4, 8
);
const pMatResult = CreateMatrix(
 1, 0, 0, 0,
 0, 1, 0, 0,
 0, 0, 1, 0,
  0, 0, 0, 1
);
MultiplyMatrix(pMat1, pMat2, pMatResult)
// [1, 5, 1, 5, 2, 6, 2, 6, 3, 7, 3, 7, 4, 8, 4, 8]
console.log(new Float32Array(Module.buffer, pMatResult, 16));
```

4.5 ベンチマーク

それでは JavaScript と WebAssembly のパフォーマンスを計測してみます*⁶。 それぞれ3秒間行列の演算を繰り返し、行列の乗算関数を何度呼び出すことができたか

^{*6} https://nyamadan.github.io/m4mul/

```
を計測してみます。
```

▼リスト 4.9 JavaScript による実装のベンチマーク

```
const ms = 3000;
let count = 0;
for (const t = Date.now() + ms; Date.now() < t; count++) {
   MultiplyMatrixJS(mat1, mat2, matResult);
}
const score = Math.floor(count / ms);
```

▼リスト 4.10 WebAssembly による実装のベンチマーク

```
const ms = 3000;
let count = 0;
for (const t = Date.now() + ms; Date.now() < t; count++) {
   MultiplyMatrix(pMat1, pMat2, pMatResult);
}
const score = Math.floor(count / ms);
console.log(score);
```

リスト 4.9 とリスト 4.10 の結果は、筆者環境の Google Chrome 65 にて JavaScript は 3380、WebAssembly は 4507 のスコアとなりました。Edge や Firefox も試してみました が同様に WebAssembly のほうがよいスコアとなりました。

4.6 まとめ

今回の実験で行列演算のようなちょっとした演算でも WebAssembly を使用したほう がパフォーマンスとしてよいということがわかりました。一方で WebAssembly で確保 したメモリは free関数で解放する必要があるため、その点は注意が必要です。

WebAssembly を使って Web の世界をもっともっと高速化していきたいですね。

やまだ

第5章

Python インスタンスの属性は辞書 (dict) で管理されているのか調べ てみた

聞いたことはあったものの、日常に影響はそうないため確認せずに済ませていたことが ありました。Python プログラミング言語において、クラスインスタンスの属性の管理に は組み込み型である辞書(dict)そのものが使われているらしい、と。

公式ドキュメント、言語リファレンスのデータモデルには次のように書かれています*1。

属性アクセスのデフォルトの動作はオブジェクトの辞書から値を取り出したり、値 を設定したり、削除したりするというものです。たとえば、a.xによる属性の検索 では、まず a.__dict__['x']、次に type(a).__dict__['x']、そして type(a) の基底クラスでメタクラスでないものに続く、といった具合に連鎖が起こります。

「デフォルトの」という但し書きがあるように、__slots__をもつクラスは辞書をもた なくなる、__getattribute__をもつクラスはこれを用いて属性の解決をおこなうように なる、など動作が書き換わることはありますが、通常は辞書で管理されているのだと読み 取れます。

そう、ここまでの理解でした。とはいえ、疑問は残ります。インスタンスの__dict__ 属性が「Python からみると辞書にみえる別のなにか」かもしれないという可能性。であ るならば、どのように実装されているものか直接見てみよう、と思い立ちました。Python の一実装、C 言語で作られた CPython のコードは公開されているのですから。

今回は Python 3.6.5*2のコードを追うことにします。

^{*1} https://docs.python.jp/3/reference/datamodel.html

^{*2} https://github.com/python/cpython/tree/v3.6.5

5.1 dis-バイトコードの逆アセンブラ

インスタンス属性がどこにどのように保持されているのかを探すのも良さそうですが、 今回はインスタンス属性に触れるコードがどのようなバイトコードにコンパイルされるの かから追ってみることにしました。

そのためのライブラリは標準で用意されています。dis モジュールです。dis.dis関数 にコード片を与えるとコンパイル、そして逆コンパイルからの解析が行われます。



これにより spam.hamというコードは LOAD_NAME・LOAD_ATTR・RETURN_VALUEという 3 つのバイトコード命令になることがわかりました。LOAD_ATTR命令の説明をマニュアル で調べると次のように書かれています^{*3}。

TOSを getattr(TOS, co_names[namei])と入れ替えます。

TOSが何なのかは置いておいて^{*4}。getattrとありこれが属性を取り出す命令であることがわかりました。そしてこれらの命令名でコードを探ると Python/ceval.c というファイルにたどり着きます。

5.2 ceval.c - バイトコードの実行処理

早速 ceval.c ファイルを開きました。先頭には"Execute compiled code"とのコメントがあります。バイトコードをどのように実行するのかが書かれたコードと踏んで読みすすめます (リスト 5.1)。

ここで PyObject_GetAttr関数が見つかりました。C 言語で Python 拡張を作る際、 オブジェクトから属性を取り出すために使う関数と同じのものです。普段のバイトコード の実行処理で使われる関数と C 製拡張でつかう関数が同じであるということにたどりつ き、Python を使うだけでなく少しは中身が読めるようになった楽しさを味わいつつ、読 み進めます。

▼リスト 5.1 ceval.c - LOAD_ATTR

^{*3} https://docs.python.org/ja/3/library/dis.html#opcode-LOAD_ATTR

^{*4} TOS は現在実行中の VM におけるスタックの先頭を指すものだそうです。つまり、事前に積まれている オブジェクトから属性を取り出して同じ位置に置き直すのですね。

```
TARGET(LOAD_ATTR) {
    PyObject *name = GETITEM(names, oparg);
    PyObject *owner = TOP();
    PyObject *res = PyObject_GetAttr(owner, name);
    Py_DECREF(owner);
    SET_TOP(res);
    if (res == NULL)
        goto error;
    DISPATCH();
}
```

5.3 object.c - PyObject_GetAttr の実装

PyObject_GetAttrの実装の一部です(リスト 5.2)。

▼リスト 5.2 object.c - PyObject_GetAttr

```
PyObject *
PyObject_GetAttr(PyObject *v, PyObject *name)
{
    PyTypeObject *tp = Py_TYPE(v);
    // 変数宣言を一部省略
    if (tp->tp_getattro != NULL)
        return (*tp->tp_getattro)(v, name);
```

PyTypeObjectの tp_getattroが指している関数を呼び出しています。NULLチェック を行っていることから可変、かつ NULLになることもあるのでしょうか。公式ドキュメン トを調べると PyTypeObject.tp_getattroそのものの説明が見つかりました*5。

オプションのポインタで、get-attributeを実装している関数を指します。シグ ネチャは PyObject_GetAttr()と同じです。通常の属性検索を実装している PyOb ject_GenericGetAttr()をこのフィールドに設定しておくとたいていの場合は便 利です。

ここに独自の関数を入れることで属性検索の方法を変えることが可能であるものの、通 常は PyObject_GenericGetAttrが収められていると推測されます。ではこれも読んで みます (リスト 5.3)。

▼リスト 5.3 object.c - PyObject_GenericGetAttr

```
PyObject *
PyObject_GenericGetAttr(PyObject *obj, PyObject *name)
{
    return _PyObject_GenericGetAttrWithDict(obj, name, NULL);
}
```

^{*5} https://docs.python.org/ja/3/c-api/typeobj.html#c.PyTypeObject.tp_getattro

なるほど、_PyObject_GenericGetAttrWithDictの dict引数に NULLを指定して実 行しています。ではこちらも (リスト 5.4)。

▼リスト 5.4 object.c - PyObject GenericGetAttrWithDict

```
PyObject *
_PyObject_GenericGetAttrWithDict(PyObject *obj, PyObject *name, PyObject *dict)
   PyTypeObject *tp = Py_TYPE(obj);
   PyObject *res = NULL;
   Py_ssize_t dictoffset;
   PyObject **dictptr;
   // 中略 tp_descr_get を探し、あればこれを使って値を取得。失敗したら次へ
   if (dict == NULL) {
       dictoffset = tp->tp_dictoffset;
       if (dictoffset != 0) {
           dictptr = (PyObject **) ((char *)obj + dictoffset);
           dict = *dictptr;
       7
   }
   if (dict != NULL) {
       Py_INCREF(dict);
       res = PyDict_GetItem(dict, name);
```

属性が記録されているオブジェクトの位置を算出し、そこから PyDict_GetItemで値 を取り出していました。事前にオプションのデスクリプタ tp_descr_getの実行があり、 属性アクセスのデフォルト処理が上書きされているケースへの対処が見られます。また、 tp_dictoffsetが0でないことの確認もあり、属性用辞書を持たないケースへの対処も 見られました。

では最後にこの dict変数の中身が本当に組み込み型の辞書なのかどうかを調べていき ます。dict変数は PyDict_GetItem関数に渡され、そこではまず PyDict_Checkマクロ が呼ばれています (リスト 5.5)。

▼リスト 5.5 dictobject.c - PyDict_GetItem

```
PyObject *
PyDict_GetItem(PyObject *op, PyObject *key)
{
    if (!PyDict_Check(op))
        return NULL;
```

このマクロで辞書かそのサブクラスであることの型確認が行われていました(リスト 5.6)。

▼リスト 5.6 dictobject.h - PyDict_Check

つまり、dict変数の中身は組み込みの辞書またはそのサブクラスであることがわかり ました。

5.4 おわりに

というわけで、インスタンス属性の管理には組み込み型の辞書(dict)またはそのサ ブクラスが使われるというところまでコードを追いかけることができました。今回読んだ 範囲では辞書そのものであるかどうかの確認はとれなかったのですが、人から聞いた、ド キュメントで読んだという段階から一歩進むことができたのではないかと思います。もう 一歩踏み込むにはインスタンスを作成する際のメモリ確保と初期化の処理を追う必要があ りそうです。

Shunsuke Ito / @fgshun


第6章

リアルの街を Unity に取り込む ~GIS へのいざない~

実際の街を広範囲にゲームに取り込みたいという要望はさまざまなところにあります。 当然まともにゼロから取り組めば、すべてのモデルを作成するという途方もない作業量が 必要になります。そのため多くの場合地図データなどから自動生成することになります。

しかしながら地図データは有料のものが多く、またゲームエンジンに簡単に取り込んで 無償で使えるものにはさまざまな制約がついていたりします。

本稿では、実際の街を国土地理院などが公開しているオープンなデータセットから、フ リーウェアや OSS を活用してさまざまなデータ処理を行い、Unity にデータとしてイン ポートするまでを解説します。(一部でシェアウェアなども使いますが、その工程につ いては OSS の同等なソフトウェアなどで代替可能です。)その過程で地理情報と呼ばれ る位置や場所を表すデータの扱い方や勘所と、地理情報を扱うための GIS (Geographic Information System) と呼ばれるソフトウェアの概念や使い方などを解説していきたい と思います。

対象として東京近辺を例に進めますが、本稿の手法は日本全国どこでも適用できると思 われます。海外の街に関しては、同様なオープンデータセットが存在するかにもよります が、工夫次第で同様に適用できると思われます。

本稿が皆さんの創造力の一助になれば幸いです。

6.1 商用、先行事例など

はじめに、商用などですでに実際の街をゲームに取り込むことのできるソリューション を紹介します。もし、これらのうちどれかを使うことができる(用途的にも経済的にも) なら、迷わず使いましょう。地理情報の処理は専門的な知識が必要になり、また大きな データセットを扱うことになります。PC のリソース要求も高いし、処理時間も長時間に わたります。本稿の執筆中も処理が終わらなくて数日かけっぱなしにしてそれでもうまく いかなかったこともありました。とにかく、使えるものがあるなら使った方がよいです。

株式会社ゼンリン 3D 都市モデルデータ

株式会社ゼンリンが提供している 3D 都市モデルデータです*1*2。

サンプルとしていくつかの街のデータが Unity のアセットストアから利用可能です。

ESRI 株式会社 CityEngine

ESRI 株式会社が提供している、3D 都市景観モデリングソフトウェアです*³。

ルールに基づいて、自動的に都市構造を生成する機能などがあります。実際の地理情報 データに基づいてルールを設定すれば、実際の街並みに近いモデルを作成できます。

mapbox

mapbox^{*4}は、カスタムマップがメインのサービスですが、幅広く地理情報を扱ってい ます。Unity 用の Asset^{*5}があり、mapbox のデータをもとにして、Unity 内に実在の街を 取り込むことができます。見た目を変えることもでき、さまざまな応用が考えられます。

WRLD

WRLD^{*6}も、同様に実際のカスタムマップなど地理情報サービスを提供しています。 こちらの UnitySDK^{*7}で、実在の街を Unity 内に取り込むことができます。見た目を変 えることもでき、こちらもさまざまな応用が考えられます。

Google Maps API for Gaming

先日(2018年3月14日)発表されたサービス*⁸です。Google Maps のデータをゲー ム内で使えるようにするものです。もちろんゲームに合わせたカスタマイズなどもできる ようです。期待が高まります。

^{*1} 株式会社ゼンリン 3D 都市モデルデータのページ:http://www.zenrin.co.jp/product/service/ 3d/index.html

^{*&}lt;sup>2</sup> 株式会社ゼンリン ZENRIN City Asset Series:http://www.zenrin.co.jp/product/service/3d/ asset/index.html

^{*&}lt;sup>3</sup> ESRI ジャパン株式会社 CityEngine:https://www.esrij.com/products/esri-cityengine/

^{*&}lt;sup>4</sup> mapbox サイト:https://www.mapbox.com/

^{*5} mapbox UnityAsset のページ:https://www.mapbox.com/unity/

^{*6} WRLD サイト:https://www.wrld3d.com/

^{*7} WRLD UnityAsset のページ:https://docs.wrld3d.com/unity/latest/docs/api/

^{*8} Google Maps APIs for games のページ:https://developers.google.com/maps/gaming/

その他

その他にも、衛星画像の販売や地図データの販売など、ニッチな領域ではありますが、 老舗から新興ベンチャーまで多くの会社があります。目的に合わせて検討してみてくだ さい。

6.2 自前でデータを用意する意味

これだけさまざまなサービスがあるのに、それでも自前で用意しようと思った物好きな あなた。考え直すなら今のうちですよ。

自前でデータを用意するのはとにかく大変ですし、やはり精度や詳細さで商用のものに かなわない点は多くあります。しかし、自前で用意したデータを基データに「東京と同じ 配置の中世風の街並み」だったり、「必要なところだけ簡素化して軽量化し、アプリバイ ナリに含める」などのような、必要に応じてさまざまな調整や修正が可能なところは、大 きな利点だと思います。また、こうした地理空間情報の扱い方を理解しておくことは、商 用のソリューションを使う際にも大きな助けになるでしょう。

6.3 地理情報の基礎知識

作業に入る前に、地理情報を扱う上での基本的なところを確認しておきましょう。

座標・緯度経度・測地系・座標系

地物(地上にあるすべてのものを表す地理学用語です)が、地球のどこにあるかを表す には、座標が要ります。もちろん皆さんは、緯度経度という表し方をご存知だと思います が、実は同じ緯度経度でも違う場所を表す場合があることをご存知でしょうか。

緯度経度は、地球を球体(正確には回転楕円体)としたときの、球面座標(これも正確 な言い方ではないですが)のふたつの角度で地物を表す座標系です。しかし、緯度経度に は重要なパラメーターとして測地系というものが付随します。測地系は、その緯度経度が どの回転楕円体を基にしたものなのかを規定したものです。測地系には、回転楕円体のパ ラメータと、座標系の原点と軸の方向、ジオイド面の三つが含まれています。これにより ある地点を正確に表すことができます。そのため、もとにする測地系が違えば同じ緯度経 度でも違う場所を表すことになるのです。

複数の測地系がある理由としては、地球は正確な回転楕円体ではなくいびつな形をしているため、各国が自分の国の地域都合や測量の歴史的経緯により、都合のよい回転楕円体を設定していました。

しかし、GPS の発展で全地球的な座標系が必要になり、WGS84 と呼ばれる GPS で用いる回転楕円体の設定に合わせた形での測地系に、各国で切り替えるようになりました。

日本でも旧日本測地系という日本ローカルの測地系だったものが、2002年に日本測地系 2000と呼ばれる GPS の用いる測地系に合わせたものに変更されました。ちなみに、旧日 本測地系と日本測地系 2000では、東京近辺では同一緯度経度で 450m ほどのずれがあり ます。

日本測地系 2011

東日本大震災による地殻変動から、2011年に国土地理院は新しい測地系、「日本測 地系 2011」を公開しました。測地系はさまざまな要因で現実に合わせて変わりま す。旧日本測地系から日本測地系 2000 への変更は、技術の発展で GPS などによ り地球上の位置を知ることができるようになったためでした。一方で、日本測地 系 2000 から日本測地系 2011 への変更は、東日本大震災による地殻変動で測量の 基準となる三角点や水準点が移動してしまったのを補正するため行われました。

地図の投影法・図法

今回は三次元で取り込むことを考えていますし、特に地球の球面の影響が大きく出るよ うな広い範囲を取り込みませんが、「地図」という二次元の形で表現するときに気を付けな ければならないのが、投影法です。地球という三次元球面を二次元に投影するため、その ままでは大きなゆがみが生じるものを、地域や用途に応じて投影方法を工夫し、実用的な 地図に表現するための図法が各種あります。代表的なものにメルカトル図法があります。

Google マップの投影法

Google マップでは、メルカトル図法を使っています。^a ただし、緯度 +/-85 度で 切り取られています。メルカトル図法では、円筒に投影しているため緯度方向が 無限の長さになってしまうため、日常では使われることのない北極と南極の領域 が切り取られています。メルカトル図法は角度が正しく描画されるため、十分狭 い範囲だけを見ると正しく形を描画できます。そのため、小さな範囲を切り取っ て使われることの多い Web 地図やスマフォの地図アプリなどでは便利な投影法と して使われています。

a https://developers.google.com/maps/documentation/javascript/maptypes?hl= ja#WorldCoordinates

GPS

GPS は、上空を周回する人工衛星からの電波を複数比較して、三角測量の原理で自分の絶対位置を計測します。このときに取得できる緯度経度は、特殊な設定がなされていなければ WGS84 という測地系を用いています。WGS84 は実用上は、日本測地系 2000 および日本測地系 2011 とほぼ同等です。

ジオイドと標高

GPS の計測した「高度」の値と測量に基づく地図の「標高」にはずれがあります。これ は、GPS での高度は測地系で定義される楕円体表面からの高さであるのに対して、地図 の標高は平均海水面からの高さとなるからです。この平均海水面にもっとも近い「重力の 等ポテンシャル面」のことを「ジオイド」といい、標高はこのジオイド面からの高さとな ります。地図の標高と GPS の高度を合わせて処理するときは気を付けなければならない 点です。

GIS

地理的な位置に関する情報を持ったデータを、総合的に編集・加工したり分析したりな どをするソフトウェアやシステムのことを指します。今回は、OSS の GIS ソフトウェア である QGIS を使っていきます。他にも、老舗の OSSGIS で分析などに長けた GRASS や、Java の GIS ツールキットライブラリの GeoTools などもあります。また、商用のも のでは ESRI 株式会社の ArcGIS が有名です。

ラスタとベクタ

地理情報にも画像のようなラスタとベクタの二種類があります。航空写真、衛星画像の ような、ピクセルごとに値をもつタイプと、道路のネットワークや、市区町村境界のよう な、幾何学形状をデータとしてあらわしたものです。それぞれラスタとベクタと分類し ます。

ベクタデータの構成

ベクタデータは、幾何学的形状を表すデータと、各形状に紐付く属性データとになって います。たとえば、都道府県のデータがあったとして、都道府県それぞれの形が幾何学的 形状のデータとして格納されます。この場合はポリゴンやエリアといった形で格納されま す。そして、それぞれの都道府県の名前、人口、面積といった情報が、属性データとして データベースのような形で紐付けられます。このふたつのデータが紐付けられていること が GIS の大きな価値となっています。

ファイルフォーマット

ラスタの地理情報を格納するのには、GeoTIFF というファイルフォーマットがよく使われます。TIFF 画像フォーマットにメタデータとして地理情報の諸元を格納したものです。一方、ベクタの地理情報は Shape ファイルという ESRI 社の開発したフォーマットが事実上の標準となっています。データ交換の際は、これらのファイルフォーマットに変換するとよいでしょう。

またデータベース関連では、PostgreSQL に PostGIS^{*9}という地理情報を扱うための定 評のある拡張があります。MySQL でも 5.6 あたりから地理情報を扱うまともな仕組みが 実装され始めています。^{*10}

Shape ファイルについて

Shape ファイルは、実際には複数のファイル郡から成り立つファイル形式です。 地理情報の幾何学的情報を格納しているのが、.shp ファイルで、.dbf が属性値を 格納しています。この.dbf ファイルは、dBASE というデータベースシステムの ファイルフォーマットです。そして、.shx ファイルが.shp と.dbf のデータを紐付 ける役割をしています。他にもいくつかの補助的なファイルがあり、それらをま とめて Shape ファイルとして扱います。基本的にはファイル名で紐付けられるの で、これらのファイル群のファイル名を変えることは、思わぬ問題の原因となるた め注意が必要です。また、Shape ファイルは地理情報データの交換用フォーマッ トとしては事実上の標準といえますが、かなり古い形式でもあることからさまざ まな点で注意が必要です。場合によっては無理に Shape にせず、使うソフトウェ アの標準形式や、PostGIS などの地理情報を正しく扱えるデータベースを用いて データ交換をするのがよいでしょう。

数値精度

地球の周囲長は約4万kmです。メートルに直すとおよそ4000000mとなり10進数 で8桁のオーダーです。正確な位置情報の処理を行うためには、小数点以下数桁までの数 値計算の精度が保証されてないと難しいでしょう。現在一般的なコンピュータの単精度浮 動小数点数は、IEEE 754で規定される23 bit の仮数部をもつ binary32 という形式です。

^{*9} PostGIS のページ:https://postgis.net/

^{*10} MySQL の地理情報データ型のドキュメント:https://dev.mysql.com/doc/refman/5.7/en/ spatial-types.html

仮数部が 23 bit なので、10 進数でおよそ 7 桁の精度しか表すことができず、1m 単位の 精度は持てません。これでは、計算のたびに誤差が蓄積してしまいます。地理座標の計算 では、原則として倍精度浮動小数点数を用いるようにしたほうがよいです。一方で、一般 的なゲームエンジンの座標値は単精度で表現されていたりもするので、データ処理時は倍 精度、最終データでは単精度など、工夫をしなければなりません。また、データ処理時に も、倍精度を使っていても、計算誤差を減らすための注意はしましょう。

地理空間インデックス

大量の地理情報データの処理には、マシンパワーが必要になります。データベース処理 では、適切に構築されたインデックスを用いて高速化することが一般的ですが、地理情報 処理でも同様に、空間インデックスを使います。四分木や、R 木などを用いて、空間内で の位置や範囲の検索を高速化します。

6.4 QGIS のインストール

さて、座学はここまでです。早速 OSS の GIS ソフトウェアである QGIS をインストー ルして、データを処理していきましょう。QGIS のサイト*¹¹から、最新版のスタンドア ローンインストーラーをダウンロードします。自分の使っている OS に合わせて選んでく ださい。本稿執筆時点での最新が 3.0.0 なので、本稿では「Windows 版 QGIS スタンド アローンインストーラ Version 3.0 (64bit)」を用います。

To get the bleeding-e	dge development build choose Advanced Install and select qgis-d	ev-full
Standalone installer	s from OSGeo4W packages	
Latest release (richest o	on features):	
± 💽 md5	QGIS スタンドアローンインストーラVersion 3.0 (64 bit)	e" e"
± 💽	QGIS スタンドアローンインストーラ Version 3.0 (32 bit)	e" e"

▲図 6.1 QGIS のダウンロード

ダウンロードしたら実行して、インストーラの指示にしたがってインストールしてくだ さい。なお、このとき指定するインストール先フォルダには多バイト文字やスペースを入 れないようにしてください。後のデータ処理でうまくいかず、余計なトラブルシュートを する羽目になります。

^{*11} QGIS のページ:https://www.qgis.org/

6.5 データのダウンロード

次に GIS データをダウンロードしましょう。今回使用するのは次の2つです。

- 国土地理院の基盤地図情報
- NTT データと RESTEC 提供の AW3D30

基盤地図情報

基盤地図情報のサイトを表示し^{*12}、その中の基盤地図情報のダウンロードをクリック します。

•	国土交通 国土 Geospatial Info	省 地理院 mation Authority of Japan	A starter	▼ 本) Go	文へ 文字サ ogle カスタム様	イズ変更 標準 1 統 (検)	大 D Engl	lish ?ップ
	地理院ホーム	国土地理院の紹介	基準点・測地観測データ	地図·空中写真	防災関連	GIS・国土の情報	申請·承認	
	地理院ホーム>	<u>地図·空中写真</u> > 基盤	也図1輛役サイト			最終更新日:2	016年5月30日	
	基盤地	図情報サイト						
	基盤地図作	青報関連ペ−ジ						
	 基盤地域 下記画像をり 基盤地 ダウン 		基盤地図情報をダウンロード	できます。過去に公	開した基盤地	2図情報もダウンロード	できます。	
	 基盤地路 下記画像を 	凶情報の登傭状況 フリックいただくと、基盤地	図情報の整備状況、整備部	範囲を調べることがで	きます。			
	基盤出 整備制	也図情報の♥ 犬況						
	各画像をクリ	ックいただくと、関連ペー	ジをご覧しただけます。					

▲図 6.2 基盤地図情報トップページ

データのダウンロードにはユーザー登録が必要なので、新規登録からユーザー登録をし ます。そして、基盤地図情報の基本項目のファイル選択をクリックします。

^{*12} 基盤地図情報サイト:http://www.gsi.go.jp/kiban/index.html

ード データ	の説明	利用者登録	各種資料	更新情報	お知らせ	利用規約	使い方	FAQ	81 1
の知らせ									
2017/03/23	機能改	良したサイトを公	閉しました。						
2017/02/28	款值課	高モデル(5mメ・	クシュ、航空レー	ザ)の「提供範囲	と作業年度」が得	認できるよう	colston 🖉		
2016/10/31	較循準	南モデルの提供テ	ータセットの文	≠⊐−F € SHIFT-	ISからUTF-8にま	更しました。			
2016/04/28	基本項	目の最新データと	過去データのダ	ウンロードサイト	と統合しました。				
								1	お知らせ
								1	お知らせ
ダウンロード								1	お知らせ
ダウンロード		604407704847 m. (w						4	お知らせ
ダウンロード ダウンロード	ドしたい基	盤地図播駅の「フ	ファイル選択へ」	ボタンをクリック	してください。				<u>8知らせ</u>
ダウンロード ダウンロード	ドしたい基	留地図情報の「J	フィル選択へ」	ボタンをクリック	してください。		其船地网	3 815 tH	<u>時知らせ</u>
ダ ウンロード ダウンロード	ドしたい基 整塑地図	留地図遺稿の「5 情報 〔日	ファイル選択へ」	ボタンをクリック 基盤地図 *** 14 博 英	してください。 情報 エーデル。		基盤地図	」 3債報 ・エデル	<u>8知らせ</u>
ダウンロード ダウンロー	ドレたい基 基盤地図	^{全地図像税の「7} 情報 [目	ファイル選択へ」	ポタンをクリック 基型地図 数値標高・	Lてください。 情報 モ <i>デル</i>		基盤地図 ジオイド	」 O情報 ・モデル	お知らせ
ダウンロード ダウンロード 通気のデー	Fしたい 基盤地図 基本項 タ6ダウン	望地図情報の「7 情報 [目	ファイル選択へ」	ポタンをクリック 基盤地図 数値標高。 あのデータはダフンロ	してください。 情報 モ <i>デル</i> aードできません		基盤地図 ジオイド	」 3情報 ・モデル	8 <u>知</u> ら5
ダウンロード ダウンロード まのデー マフ	Fしたい 基盤地図 タもダウン アイル)	 金地図情報の「7	7アイル選択へ」	ボタンモクリック 基盤地図 数値標高。 ^{まのデータはダフンロ} ファイル	してください。 情報 モデル ==ドできません 波訳へ		基盤地図 ジオイド・ ファイル	3 20債報 ・モデル 20歳れへ	<u>8知ら</u> 1
ダウンロード ダウンロード 通知のデー マ	Fしたい基 基型地図 タもダウン アイル道	 金地図情報の「方 情報 目 Bードできます 	7アイル選択へ」	ボタンモクリック 基盤地図 数値標高・ まのデータはダクンド ファイル2	してください。 情報 モデル ==ドできません 波沢へ		基盤地図 ジオイド ファイル	⊴ 価情報 ・モデル 選択へ	お知らせ

▲図 6.3 基本項目ファイル選択

日本地図が表示されました。ここから必要な項目と地域を選択します。今回は「建築物の外周線」のみにチェックを入れます。

また、地域は自分の馴染みのある地域の方が作業がしやすいですが、適当に選びましょう。例として、港区を含む東京の 533945,533946,533935,533936 の4 区画を選びます。 「ダウンロードファイル確認へ」をクリックします。



▲図 6.4 基本項目データ選択

「全てにチェック」して、「まとめてダウンロード」します。

10				<u> <u> </u> <i> </i></u>	基础地図情報	<u>R#416</u>	<u>地理院ホーム</u>
***		#/~####/##/ ##### #		5 11 + +			
	- 後に多くの選択を行うと、テーヌ	ティスが大きくなり、メウノロート	てきないことか	カリより			
展る	トップページに戻る						
E w ay de		alfa					
	1(7177 1) + CU(7777	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1	1		-	2
チェック	ファイル名	基盤地図情報種別	更新年月日	項目分類	項目名	存重 (KB)	個別
	EG.GMI -533935-11-20180101 via	其般地図情報 母野データ	2018年01日01日	533935	建築物の外	58643	ダウンロート
	TO ONE SSSSS IT EDIOLOGICA	SECURIE W AGE /	2010-01/3011	000000	周線	00040	(ログインが必要で
	FG-GML-533936-11-20180101 via	其效地回情部 番野平一々	2018年01月01日	522926	建築物の外	4993	ダウンロート
æ	PG-GML-555556-11-20160101.20		2010-01/5018	333330	周線	4005	(ログインガル要で
	EC. CHI . 5330/5-11-20180101	草段边面建筑 美公本一方	2018年01月01日	532045	建築物の外	64640	ダウンロート
. .	PG-GML-555945-11-20180101.20	登显地国间和 取利7-7-7	2010+01/010	333843	周線	04043	(ログインが必要で
	FC-CML 522046-11-20180101-10			E22046	建築物の外	50510	ダウンロート
- M	PG-GML-555946-11-20180101.20	登里地區16款 款約7一次	2010-01/018	533940	周線	52510	(ロゲインが必要で
	学会を対象の東京派制品技術は、一	申請(ワンストップサービス)用入力				1	ダウンロート
-	务23元为重成未少该会未动中的xmi	補助ファイル					(ログインが必要で
						1	ダウンロート
	第20条测导式用小使用表现中期;;;;;)	申請(ワンストップサービス)用入力					

▲図 6.5 基本項目ダウンロード確認ページ

これで、選択したデータの格納された Zip ファイルがダウンロードされます。 ダウンロードのページまで戻り、同様に数値標高モデルをダウンロードします。

9-F	7-*	の説明	利用者登録	各種資料	更新情報	お知らせ	利用規約	使い方	FAQ	お問い
お知ら	t									
201	17/03/23	機能改良	見したサイトを公	閉しました。						
201	17/02/28	政任課	長テル (5mメ・	シュ、航空レー	サ)の「提供範囲	と作業年度」が用	認できるよう	になりました。		
201	16/10/31	數值標訊	あモデルの提供デ	ータセットの文	≠⊐−ドをSHIFT-	JISからUTF-8に家	更しました。			
201	E /04/28									
ダウン	- F	基本項目	■の最新データと	過去データのダ	ウンロードサイト	を統合しました。			3	<u>お知らせ</u> —】
ダウン		基本項[したい基]	■の最新データと 留地図講報の「フ	過去データのダ 'アイル選択へ」	ウンロードサイト! ポタンをクリック	を統合しました。				8知らせ - 】
ダウン ダウ	イロード インロード 基	基本演 したい基 盛地図f	10 最新データと 留地図情報の「フ 育報	透去データのダ ァイル選択へ」	ウンロードサイト* ボタンモクリック 基型地図	を統合しました。		基盤地図	请银	81116世—3
ダウン	マード フンロード 基 上	基本項目 したい基料 盛地図作	10最新データと 留地図情報の「フ 育報 目	透去データのダ アイル選択へ」	9ンロードサイト1 ボタンモクリック 基盤地図 数値標高・	を統合しました。 してください。 酒報 モデル		基盤地図 ジオイド・	情報 モデル	お知らせ一貫
ダウン ダウ	vロード フンロード 基 またのデータ	基本項目 したい基 盤地図1 5本項	100最新データと 論地回情報の「フ 育報 日 →-ドできます	過去データのダ ァイル選択へ」 通	ッンロードサイト1 ボタンモクリック 基型地団 数値標高: まのデータはダンン1	E就会しました。 UT<ださい。 I情報 モデル ロードできません		基盤地図 ジオイド・	情報 モデル	<u>お知らせ</u> ー】
ダウン ダウ	マード フンロード 基 まのデータ	基本項目 したい基	100最新データと 20世国情報の「フ 育報 日 	過去データのダ ァイル選択へ」 通	9ンロードサイト1 ボタンモクリック 基盤地図 数値標高・ 50データはダラン1	を統合しました。 してください。 情報 モデル ロードできません		基盤地図 ジオイド・	情報 モデル	お知らせ―5
ダウン	マロード ウンロード 基 出来のデータ フィ	基本項目 したい著 登地図作 も本項 ドレダウンド	100最新データと 20世国情報の「フ 育報 日 	過去データのダ ァイル選択へ」 達	9ンロードサイト1 ボタンモクリック 基登地図 数値標高。 50データルダウンド ファイル3	を統合しました。 してください。		基盤地図 ジオイド・ ファイル3	情報 モデル Ĕ訳へ	お知らせ―5

▲図 6.6 数値標高モデル選択

こちらも同じ地域で 5m メッシュを選択してください。同様に地図で選択して、ダウン ロードします。



▲図 6.7 数値標高モデルデータ選択

News				<u> 8777</u> <u>8</u>	堂地図慣報	<u> </u>	地理院ホーム
277							
<u>*</u> .	一度に多くの選択を行うと、データ	サイズが大きくなり、ダウン	ロードできない	ことがあります			
戻る	トップページに戻る						
P. S. Prose							
- x 7	全でチェック まとめてダウンロ	a — F Alle					
		-	-			容量	#104
· = 77	ファイル名	基盤地図情報種別	更新半月日	項目分類	項目名	(KB)	個別
	FG-GML-5339-35-DEM5A.zip	基盤地図情報(数値標高モデ	2016年10月01日	5mメッシュ (標高)	5339-	6384	ダウンロード
		ル)			35		(ログインガ会装で)
	FG-GML-5339-36-DEM5A.zip	基盤地図情報(数値標高モデ ル)	2016年10月01日	5mメッシュ (標高)	5339-	3319	ダウンロード
		其段纵回得起(於信標室本平			5220-		WALL I
2	FG-GML-5339-45-DEM5A.zip	IL)	2016年10月01日	5mメッシュ (標高)	45	6221	タウンロート (ログインが必要で
		基盤地図情報(数値標高モデ	0017/00/00/00/0		5339-		ダウンロード
×	PG-GML-5339-46-DEM5A.2ID	N-)	2017#02/3028	5mメッシュ ((原金)	46	5341	(ログインが必要で)
	fmdid15+3101 vml	メタデータ		5mメッシュ (標高)	DEMSA		ダウンロード
	Internal Protection			(航空レーザ測量)	DEMOR		(ログインが必要で)
	第29条測量成果の複製金額申請.xml	申請(ワンストップサービス)					ダウンロード
		用入力捕助ファイル					(ログインガ必要で)
	THE REPORT OF THE REPORT OF	申請(ワンストップサービス)					ダウンロード
	第30〜測算広坐の使用本認単語.xm						

▲図 6.8 数値標高モデルダウンロード確認ページ

最後に、もう一度ダウンロードのページに戻り、「符号化規則、ファイル仕様書、表示 ソフトウェア等」のリンクから、

数値標高モデル ^{のデータはダウンヨードできません}	ジオイド	・モデル
のデータはダウンロードできません		
ファイル選択へ	ファイル	選択へ
<u>データの説明</u>	<u>7-20</u>	11HO
いたします。		
の申請が必要となる場合があります	。詳しくは <u>利用規約</u> をご覧くた	tëv.
	ファイル選択へ データの説明 r。 いいたします。 同の申請が必要となる場合があります	ファイル選択へ <u>データの説明</u> ア。 いいたします。 同の申請が必要となる場合があります。詳しくは <u>利田規約</u> をご覧くれ

▲図 6.9 表示ソフトウェア等リンク

「基盤地図情報ビューア」をダウンロードします。

表示ソフトウェア
<u>基盤地図情報ビューア</u> ZIP形式:6.85MB 2018/03/14 更新 重や頃 日で奴偃傳商モデルの表示ソフトウェア
Shape形式、拡張DM形式等へのエクスポートも可能です
※簡易的な表示ソフトウェアのため、大量のデータの表示・エクスポートはできません。
※操作がうまくいかない場合は、データを分割して処理してください
<u>基盤地図情報ビューア操作説明書</u> PDF形式:2.1MB 2018/03/14 更新

▲図 6.10 表示ソフトウェアダウンロードリンク

AW3D30

AW3D30のサイトを表示します^{*13}。こちらでもユーザー登録が必要です。上記ページ の「4. Download」の項目から、ユーザー登録とダウンロード用のサイトへのリンクがあ ります。世界地図から何度か目的の地域にドリルダウンします。

^{*13} AW3D30 のサイト: http://www.eorc.jaxa.jp/ALOS/en/aw3d30/



- 利用规约 / Terms of use

▲図 6.11 世界地図表示画面

地図から必要な地域を選び「Download」します。



▲図 6.12 AW3D30 ダウンロード画面

5×5の領域を一括でダウンロードすることもできるので活用します。今回の例として、N035E139の名前のついているデータを使用しますので、例にしたがって試してみる

方はこのデータをダウンロードしてください。なお注意点として、ここでは FTP でダウ ンロードするため、ファイヤーウォールなどの設定を見直す必要があるかもしれません。

ダウンロードしたファイルは tar.gz 形式の圧縮ファイルで、GeoTiff 形式の DSM が収 められています。DSM とは Digital Surface Model の略で、建物や樹木などの地物を含 めた地表面の高さのデータです。

6.6 データの変換

基盤地図ビューア FGDV によるデータの確認と Shape ファイルへの変換をします。ダ ウンロードした基盤地図情報ビューアを展開すると、FGDV.exe という実行ファイルが 得られます。これを実行して図 6.13 のようなソフトウェアを起動します。



▲図 6.13 基盤地図情報ビューア

次に、左上の白い四角のアイコンまたは「ファイル」メニューから、「新規プロジェクト作成」を選びます。新規プロジェクト作成ダイアログが出てくるので、「追加」で先ほ どダウンロードしたファイルを選択します。基盤地図情報ビューアは、Zipファイルのま まで読み込めるので、Zipファイルを指定します。

🥮 基盤地図情報ビューア		_		\times
ファイル(F) 設定(L) 表示(V) 属性(A) 言	計測(R) エクスポート(E) ヘルプ(H)			
	▋□ੳQ⊡፼፼₽¼I₿≌ 』₿₽₭₺	12	2 द	Sen.
	基盤地図情報ビュ またまれし図作音報ビュ ままたまれし図作音など ままたまれしの作音なの ままたしての作成 あ込むアイル ※型地図プジェクトでのがした339-35-DEM5A2p ¥data¥eiedata2¥FG-GML-5339-35-DEM5A2p ¥data¥eiedata2¥FG-GML-5339-45-DEM5A2p ¥data¥eiedata2¥FG-GML-5339-45-DEM5A2p ¥data¥eiedata2¥FG-GML-5339-46-DEM5A2p ¥data¥eiedata2¥FG-GML-5339-46-DEM5A2p ¥data¥eiedata2¥FG-GML-5339-46-DEM5A2p ¥data¥eiedata2¥FG-GML-5339-46-DEM5A2p ¥data¥eiedata2¥FG-GML-5339-46-DEM5A2p		ersion のK キャンセル 解除 余) で で 解除 余) で で で で で で で で で で で で で	

▲図 6.14 新規プロジェクト作成

それなりに時間がかかりますが、うまくいけば、次のような画面が表示されます。



▲図 6.15 建築物の外周線データの表示



▲図 6.16 数値標高モデルデータの表示

次に、データを Shape ファイルにエクスポートします。 建築物の外周線のデータは「エクスポート」メニューから「エクスポート」を選びます。



▲図 6.17 建築物の外周線データをエクスポート

出てきたダイアログの「変換種別」を「シェープファイル」に、「建築物の外周線」の チェックを外し、出力先フォルダを適切に設定して「OK」を押します。

エクスポート	×
変換種別 シェープファイル ~	
変換する要素	変換する領域 それンセル
20180101	● 全データ領域を出力
	○ (おおむね)現在表示されている要素のみを出力
-	○設定された領域内の要素のみを出力
	○ (全データを)DM図郭単位に出力
	○(おおむね)現在表示されているDM図郭単位に出力
	○設定された領域内のDM図郭単位に出力
	○選択されたDM図郭単位に出力
すべてON すべてOFF	র্বনের্টেম রুর্বেCOFF
出力先フォルダ D:¥data¥国土地理院2¥BLD4	

▲図 6.18 データエクスポートダイアログ

特に大都市の中心部では建物の数が多く、「Out of Memory」などのエラーが起こる 可能性がありますので、適切に一区画ごとに読み込んでエクスポートするなどしてくだ さい。

数値標高モデルは「エクスポート」メニューから「標高メッシュをシェープファイルへ 出力」を選びます。



▲図 6.19 標高メッシュをエクスポート

出てきたダイアログの出力先ファイルだけ適切に調整して「OK」を押します。なお、 ファイルのパスにはマルチバイト文字やスペースを含まない方が余計な問題が起きなくて 楽です。

標高メッシュデータのシェープファイルデータへの変換	×
□直角座標系に変換して出力 9系 ✓	🖌 ок
● 全データを出力	🗶 キャンセル
○ (おおむね)現在表示されている要素のみを出力	
○ 設定された領域内の要素のみを出力	
出力先ファイル	出力サイズ確認
D¥data¥国土地理院2¥dem.shp	

▲図 6.20 標高メッシュエクスポートダイアログ

もし「Out of Memory」のエラーが発生したら、一区画ごとにエクスポートしてくだ さい。

以上の手順で、建築物の外周線の Shape ファイルと数値標高メッシュの Shape ファイ ルが作成できました。

6.7 データの読み込みと QGIS の基本操作

「QGIS Desktop 3.0.0 with GRASS 7.4.0」を起動してください。起動したらまず、プ ロジェクトを作ります。「プロジェクト」-「新規作成」を選んでください。空のプロジェ クトが作成されるので、メニューの「プロジェクト」-「保存」によりファイル名等を指定 して保存します。

次に、「レイヤ」-「レイヤの追加」-「ベクタレイヤの追加」を選択します。



▲図 6.21 ベクタレイヤの追加

表示されるダイアログで、先ほど基盤地図情報ビューアでエクスポートした建築物の外 周線データの Shape ファイルを指定して「追加」を押します。

Q データソースマネージャー ベクタ					?	×
🗁 דאָליד 🏫	ソースタイプ					
V ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	● ファイル(1)	○ ディレクトリ(D)	○ データベース(I)	O プロトコル(L)		
	エンコーディング		UTF-8			•
578	ソース					
ラੵੵテリミティッドテキスト	ベクタデータセット [):¥data¥gisdata2¥BLD3¥20180101-建築	物shp		•	•
🤗 GeoPackage						
🍂 SpatiaLite						
PostgreSQL						
MSSQL						
Oracle						
DB2 DB2						
₩ 仮想レイヤー						
C wms						
🚓 wcs						
E WFS						
ArcGIS Map Server				Close ittn(4)	Li-le	
ArcGIS Feature Server 🗸 🗸					neip	,

▲図 6.22 ベクタレイヤの追加ダイアログ

図 6.23 のように読み込まれます。



▲図 6.23 ベクタレイヤが読み込まれたところ

こうして建築物のデータを読み込みました。同様の手順で数値標高のデータも読み込み ます。数値標高のデータは 5m 間隔の格子状に配置された点のデータになるので、かなり 描画の量が増え、重くなるので気を付けてください。

次に、「レイヤ」-「レイヤの追加」-「ラスタレイヤの追加」を選択します。

表示されるダイアログで、ファイル名を設定して「追加」を押して、AW3D30のラス タデータを読み込みます。

Q データソースマネージャー ラスタ		? ×
🃁 דָאָלָד	ソース	
V, ~*>>	ラスタデータセット D+#data¥3GSDATA_SAMPLE¥AW3D30DSMNN035E189_AVE_DSM.tif	
7 ,9		
ラੵੵテリミティッドテキスト		
🥰 GeoPackage		
🍂 SpatiaLite		
₩_ PostereSQL		
MSSOL		
📮 Oracle		
DB2 DB2		
₩ 仮想レイヤー		
💮 wms		
🚑 wos		
WFS		
ArcGIS Map Server		11-1-
ArcGIS Feature Server 🗸 🗸	Close (1970(A)	Help :

▲図 6.24 ラスタレイヤの追加ダイアログ

最後に、QGISの基本操作ですが、手のアイコンはドラッグで地図の移動になります。 スペースキーを押しながらドラッグすることで、地図の移動をすることもできます。ま た、マウスのホイールまたは「+」「-」の虫眼鏡アイコンで拡大縮小ができます。

6.8 データの処理 その1

一般的な GIS ソフトウェアには、単に地理情報データを表示するツールだけでなく、 さまざまなアルゴリズムで地理情報データを処理したり分析するツール群も含まれていま す。ここからはこれらのツール群を駆使して、用意したデータから建物の高さを計算して 3D にすることをやってみましょう。

最初に、ここで行う操作の簡単な解説をします。今回用意したデータは、測量等で作成 された基盤地図情報の建築物の外周線のポリゴンデータ、レーザー測量や写真測量で作成 された基盤地図情報の数値標高モデル(以降 DEM)、衛星画像から作成された AW3D30 のデータ(以降 DSM)の3点です。DEM は、地物を除いた「地表面」の高さデータ、 DSM は、地物込みの表面の高さデータなので、その引き算で、地物の高さを求めること ができます。これを利用して、高さを持たない建築物の外周線のポリゴンに高さ情報を付 加します。DEM は、5m メッシュ、DSM は 30m メッシュなので、それなりに誤差が生 じますが、「当たらずも遠からず」のレベルでのリアルさを目指します。

準備として、複数に分かれているデータを結合します。そして最初に、ベクタデータの DEM とラスタデータの DSM を処理して高さデータを作成しましょう。

今回の処理のその他の誤差要因

さらに細かい話ですが、DSM データは ALOS という人工衛星で撮影した画像を もとに、ステレオ視の原理を用いて複数の画像から高さを測定しています。対し て、DEM データは、航空機からのレーザー測量で標高データを測定しています。 また、仕様書を確認すると DSM データは EGM96 というジオイドモデルを使用 しています。DEM 側には国土地理院の測地成果であることから、水準測量での標 高値および日本のジオイドモデルを使用していると思われます。このため、この ような計測方法や基にするジオイドモデルの違いでも誤差が生じる可能性があり ます。

建築物の外周線データの結合

基盤地図ビューア FGDV によるエクスポート時に、4 つに分けて書き出した Shape ファイルを結合します。メニューから、「ベクタ」-「データ管理ツール」-「ベクタレイヤ の結合」を選択します。



▲図 6.25 ベクタレイヤの結合

ベクタレイヤの結合ダイアログでは、入力レイヤに結合したい建築物の外周線データの レイヤをチェックを付けます。変換先 CRS は、測地系です。これは、EPSG:4326 を選 んでおきましょう。結合されたデータの保存先には、適当な名前の Shape ファイルを指 定します。そして、「バックグラウンドで実行」を押します。



▲図 6.26 ベクタレイヤの結合ダイアログ

処理が終わると新規レイヤとしてプロジェクトに追加されます。他の建築物の外周線 データのレイヤは今後必要ないので、「レイヤ」ペインで右クリックして削除しておきま しょう。

もし、DEM データも分割して書き出していた場合、同様の手順で結合しておいてください。

ポイントデータにラスタからの値を取り込む

DEM と DSM は、ベクタとラスタで種類が異なるので、そのままでは演算できません。 そこで、どちらかに合わせることになりますが、ここではより詳細な DEM の方に、DSM のデータを合わせましょう。

まず、ラスタの値を読み込むためのカラムを作成します。DEM のレイヤをダブルク リックしてレイヤプロパティのダイアログを開きます。

せっかくなので、ここで測地系の設定も行っておきましょう。左側のペインから 「ソース」を選んでください。表示された画面で、座標参照系を設定します。ここでは、 EPSG:4326 を設定しましょう。これは、WGS84 を意味します。

また、データソースエンコーディングで文字列のエンコーディングを選べます。今回は System にするとよいでしょう。もし、カラム名などが文字化けしているときはここの設 定を見直してみます。

Q レイヤプロパティ - dem ソース		?	×
Q	▼ 設定		
(1) 情報	レイヤ名 dem 表示名 dem デーかリースていコーディング Swetem マ		
🗞 ג-ע	> // / / / / / / / / / / / / / / / / /		
💉 シンボロジー	ソース座標参照系を設定する		
	EPSG:4326 - WGS 84	-	-
(abc) ラベル	空間インデックスの作成 領域の更新		
🐪 ダイアグラム	▼ プロバイダの地物フィルタ		
🔶 3DE1-			
🥛 У-274-ИК			
🔡 属性フォーム			
• 📢 結合		1	
創 補助記憶装置		ofne)	19
🔊 דבלל 🔯			
	スタイル ▼ OK Cancel Apply	He	lp

▲図 6.27 レイヤプロパティ・ソース

次に左側のペインから「ソースフィールド」を選んでください。上に並ぶアイコンの鉛 筆アイコンをクリックして編集モードにします。

Q レイヤブロパティ - dem ソースフィールド	?	×
 (i) 指報 10 名前 別名 タイフ タイフ名 長さ 精度 コメント WMS WFS 		
😻 シンボロジー		
Cate 5×04		
🍕 #17/51.		
30ピュ−		
レースフィールド		
• ◆ 結合		
(1) 注意的記憶装置		
🎸 ນວຍຫວອ		
交 支款		
ال کې ۲۰۰۶ کې	He	lp

▲図 6.28 レイヤプロパティ・ソースフィールド

編集モードにしないとカラムの変更やジオメトリの修正といったデータ編集ができない ようになっています。編集モードになると「新規フィールド」ボタンが押下可能になるの でクリックし、フィールド追加ダイアログで必要な項目を入力し、「OK」ボタンを押しま す。長さ・精度は、浮動小数点のデータの桁数設定となります。今回は高さの値なのでそ れに合わせて十分な長さを確保しておきます。

🔇 フィールドをi	追加	?	Х
名前(<u>A</u>)	dsm		
אלאב	DSM Height		
タイプ	小数点付き数値	ē(real)	-
プロバイダタイプ	double		
長さ	10		-
精度	4		-
	OK	Cano	el

▲図 6.29 フィールドの追加ダイアログ

ついでに、DEM データの「標高」フィールドがマルチバイト文字のフィールド名なの で、「dem」などと変えておいてもよいかもしれません。

最後に、再び鉛筆アイコンをクリックしてデータの変更を保存して「OK」ボタンを押 してダイアログを閉じます。

1

さて、準備が長くなりましたが、ラスタの値を読み込みます。「プロセッシング」メ ニューの「ツールボックス」を選択します。右側に新しいペインが表示されました。

ブロセッシングツールボックス		ť
🎭 🐴 🕓 📄 🔦		
Search***		
> 最近使用された		
> 🝳 グラフィックス		
> 🝳 データベース		
> 🔇 デーダ補完		
> 🔇 ネットワーク解析		
> 🝳 ファイルツール		
> 🔍 ベクタージオメトリ		
> 🔍 ベクター作成		
> 🔇 ベクター選択	Chb_Pet	
> 🔍 ベクタオーバーレイ	CON TRAC	
> 🔍 ベクタジオメトリ		
> 🝳 ベクタテーブル		
> 🔍 ベクター般		
> 🔍 べり夕解析		
> 🔍 べり外作成		
> 🔇 ラスタツール		
> 🝳 ラスタ地形解析		
> 🝳 ラスタ分析		
> 🝳 レイヤツール		
> 🔇 地図製作		
> 🔇 補間		
> 🚠 GDAL		
> 💥 GRASS		
🍫 モデル		
🥐 スクリプト		

▲図 6.30 ツールボックス

「Search」と表示されているフィールドに「v.what.rast」と入れてください。「ベクタの 重心位置でのラスタ値をテーブルにアップロードします.」という項目が見つかりますの で、これを選択します。

プロセッシングツールボックス	5 ×
🍁 🛃 🕓 🖹 🔧	
v.whatrast	0
▶ 最近使用された	
🧼 v.whatrast - べりタの重え	し位置でのラスタ値をテーブルにアップロード・・・
 ✓	
💊 v.whatrast – べりタの	重心位置でのラスタ値をテーブルにアップロ・・・
	v.what.rast - ベクタの重心位置 でのラスタ値をテーブルにアップロー ドします.
	Algorithm ID: ' grass 72 what rast

▲図 6.31 ベクタの重心位置でのラスタ値をテーブルにアップロードする

これは、ベクタデータの属性カラムにその座標でのラスタの値を取り込むツールです。 ダブルクリックして実行してみましょう。

開いたダイアログで必要項目を設定していきます。「Name of vector points map for

which to edit attributes」には、ラスタからデータを読み込むレイヤを選択します。今回の場合は、DEMのレイヤを選択します。

「Raster map to be sampled」には、読み込むラスタのレイヤを指定します。「Input feature type」では、今回の DEM データは点の集合なので、「point」を指定します。「Name of attribute column to be updated with the query result」には、先ほど新規で 作成したフィールドの名前を設定します。このフィールドにラスタからの値が読み込まれ ます。「v.out.ogr 出力タイプ」は auto でもよいですが、一応明示的に point を指定して おきます。「Sampled」は出力結果をどうするかの指定です。入力フィールド横の「…」ア イコンで「Save to File...」を選んで、適切なファイル名を設定します。

すべての設定が終わったら「実行」ボタンをクリックします。処理が終わるまでにかな りの時間がかかるので、ゆっくり待ちましょう。

なお、処理画面でコマンドラインらしき表示がたくさん出てきます。これは、GRASS という OSS の GIS ソフトウェアを呼び出しています。

GRASS は、定評のある老舗の機能豊富な GIS ソフトウェアですが、コマンドライン 中心のインターフェースから難易度が高くなっています。QGIS は、それ自体でも多くの GIS 処理用の機能を持っていますが、このようにフロントエンドとして GRASS と連携 することで、より多くの機能をより簡単に使えるように構成されています。

実行が終了すると、作成されたベクタレイヤが追加されます。レイヤプロパティから空間インデックスを作成しておくと、今後の作業が早くなるかもしれません。

最後に実行結果を確認して次の作業に移りましょう。iの文字を矢印が指しているアイ コンをクリックします。これは、地物の情報表示機能です。

DSM の値を取り込んだ結果のベクタレイヤを選択し、地図上の任意のポイントデータ をクリックします。すると、右側に地物情報のペインが開き、もともと持っていた dem のデータ値と取り込んだ dsm ラスタデータの値が格納されているのが分かると思います。

98



▲図 6.32 地物情報表示

フィールド計算機を使う

DSM の高さの値を属性に取り込んだので、元々もっていた DEM の高さと引き算をし てみましょう。この操作にはフィールド計算機を使います。アイコンバーの中のそろばん のアイコンをクリックします。でてきたダイアログで、DSM-DEM の計算を実行します。 「新しいフィールドを作る」にチェックを入れ、出力フィールド名に「height」出力フィー ルドタイプを「小数点付き数値」にします。出力フィールド長は、10、精度を4くらいに しておきましょう。「フィールドと値」の dsm をダブルクリックし、式テキストフォーム に入れます。演算子ボタンの「-」をクリックし、dem をダブルクリックして式「"dem" -"dsm"」を作ります。最後に、「OK」ボタンを押して実行します。これで、DSM-DEM で 計算した高さ情報の演算結果が新規フィールドに保存されました。



▲図 6.33 フィールド計算機を使う

地物の情報表示機能を使って、計算した結果が入っているか確認してみましょう。

最後に、フィールド計算機の操作は編集操作となるので、編集モードに自動的になって います。メニューの下のアイコンの2行目の鉛筆アイコンをクリックして編集モードを 保存して終わりましょう。と言いたいところなのですが、なぜか保存処理がとても時間 がかかるようなので、編集モードを解除せずに、レイヤペインで右クリックして、「Save as...」を選択し、図 6.34 のように形式を Shape ファイル、ファイル名に適切なもの指定 して「OK」ボタンを押して保存してください。

🔇 ベクターレ	ノイヤーを名前で保存				?	×
П< -+	FODI Ohana (ile					-
7514	ESRI Shapetile					•
ファイル名	D:¥data¥GISDATA_SAMPLE¥DemD)smHeight.shp				
レイヤ名						
CRS	EPSG:4326 - WGS 84				•	
ד-בעד		UTF-8			•	^
選択地	也物のみ保存する					
☑ 保存さ	れたファイルを地図に追加する					
▶ エクス	ボートするフィールドとエクスボートオ	ブションの選択				-
▼ ジオメ	449					
ジオメトリ	ゆイブ	自動			-	
	チタイプにする					
Z)欠;	元を含める					
	県町(3944年・12月17)					
▼ 117						
RESIZE	NU				•	
SHPT					•	
▼ カスタ	ムオプション					
データン	-2					
117						
						~
			OK	Cancel	Hel	Þ

▲図 6.34 フィールド計算機結果をファイルに保存

元の DEM ファイルは、編集モード終了時に、保存しない「Discard」の方を選び、レ イヤリストから削除しておきましょう。

ポリゴンデータにポイントデータの値を取り込む

ここまでで、DSM と DEM から計算した地物の高さの値が各ポイントに格納されました。次に、建物外周線ポリゴンのデータに、地物の高さの値のポリゴン内での最大値を取り込んでみましょう。

ツールボックスペインで、「v.vect.stats」を検索します。見つかったらダブルクリック してツールを起動します。設定する項目は、「Name of existing vector map with points」 に、地物の高さまで計算した DEM レイヤ、「Name of existing vector map with areas」 に、建物の外周線データのレイヤを、「Method for aggregate statistics」には、領域内 に複数の点が含まれる際に集約する時の計算を指定しますが、maximum を設定します。 「Column name of points map to use for statistics」には、地物の高さが入っている フィールドの名前を指定します。「Column name to upload points count」には、領域に 含まれる点数を格納するフィールド名、「Column name to upload statistics」には、集約 した結果の値(この場合は、領域に含まれる点の height の最大値)を格納するフィール ド名を指定します。最後に、「Updated」の入力フィールドに、「…」ボタンから、「Save to File…」を選んで、保存するファイル名を指定し、「実行」ボタンを押してください。

	P Questing Mithanist Hittig at	7	×
OV.CON/		1.2	~
	103-9- 07		201441-50-54-60
	Name of existing vector map with points		
	DemOsinfleight (EPSO 4326)	•	
C BRULAD	Tione of existing vector map with areas		pearly"
*_1	Duilding (LPSQ 4121)		 Vpravel - X/P/2221 Al/T1193110 veste - Récommune / Premo Ale / Voltage / Voltage
01	Rout feature type		マクロビロー からい・シンマンションをあるしたり、ア・シンマンシー シンロロシー (日本55)、ジント しまので、ダリーたら)東洋動
DW	「「「東非道訳行はした		「「「「「「「」」」、「「「」」」、「「」」、「「」」、「「」」、「」」、「」
EN	Method for apprepate statistics		2 vacant - K00-#CUHI26090-FHT-60C/Fe02.
GW	maximum		 vrankes - 5ンダムに20 / 30パ559ーボイント地団を生成しー
OW	Column name of points map to use for statistics		vraststats - ベクトルボリエンに基づいてラスタマップから単実…
CeoPackage	A 1.3 height		 vreclass - SQLウエリの私記集または開始テーブルの序的が動に…
/ SpatiaLite	Column name to upload points count (integer, created if doesn't exists)		☆ vzectily - 新聞市に憂少いてベクトル内の日オブジェクトの寝ーー
PostQ25	Count		vreport - べうトルのジオメドリ統計を解告します。
MSSQL	Column name to upload statistics (double, created if doesn't exists)		vsample - ベラターボイントの位置でラスターレイヤーをサンプー
Chacle	E beat		↓ vseenert - 入力べうトルラインと位置からボイント/セクメント・・・
062	- Returney		☆ vselect - 他のペクターマップ(E)の地物によってペクター地図…
@ WMS	COASE OF 7 MARTINE year was used [HTEn']		
🕐 XYZ Tiles	THE-HORE VANGER DOMAGE TO SEE		 vsurtbspine - Tykhonov正則によるバイキュービックまた…
G WCS	vinogrスナップ許容証著(-1×スナップな(_)		☆ vautide - 注記職二乗重み付けによるべりトル点デー対応・・・
db wrs	× -100000		 vautat - ベラタボイントマップからスプラインで曲面補助を実一
47- 8	x vince R-188		v10.3d - 20ペウター絶物の30への実験を実行します。
* 店 米 〒 石・耳 雪 口	0.00100		 vtoSines = ベクトルボリゴンまたは点を除こ実現します。
2 Duilding			where a state of the s
DemDumHeight			vitovait - <391-1743331-1712(\$181.021)
V NOSE139_AVE_DSM		-	vitaniform - 10001110,7040/KIREAITED.
	U/OMA/GOOM IN DWARE CONTRACTING		Vide - VIXIORROVI /EXECUTE
	アトコバムの南行後に出力フィイルを開く		↓ vinion · 病性の単文量統計を計量します。分割と標準量…
			● vivectatata = 地域加点就を研え、統計を計算します。
	2		▼ VN00001 * 点を含む入力べうターレイヤーからオロノイ回を19…
	8		※ va#atrast - ペクタの重く位置でのラスタ値をテーブルにアナー
			vnhatvect = ベクトルボイントの位置にあるベクトル値をテーブ・・・
			> FIRE (NVE)
			> 画像()*)
			• E7/h
	4	OK BUCK	• A2171
		-	SAGA
	Run as Batch Process**	運行 Close Help	

▲図 6.35 v.vect.stats の実行

この処理も非常に時間がかかりますが、終わったら地物の情報表示機能で正しく集計が 行われたか確認してみましょう。

6.9 データの処理 その2

ここまでの作業で、もともと高さデータを持っていなかった建築物の外周線データに、 DSM と DEM から計算した高さのデータを持たせることができました。ここから、Unity に取り込む方法を考えてみましょう。

そもそも、ベクタデータは仕組みとしてはゲームエンジンで描画のために扱うジオメト リの仕組みと似たようなものです。そのため、原理上はベクタデータの座標を何らかの形 で Unity 上の座標に変換し、点同士の接続データなどを追加して読み込めるような形式に コンバートするとよいように思われます。

しかし、たとえば QGIS で出力可能なベクタデータ形式として DXF に出力し、試しに CAD ソフトやモデリングソフトに読み込んでみると、本稿で例として扱っているデータ などはそのままではデータ量が多すぎるため操作できない状態になります。扱えるように するには、より小さい領域で分割するなどの方法が必要そうです。

そこで、本稿では Unity 上で目で見える形にするために、ベクタデータを一度ラスタ データに変換し画像データとしたのちに、その画像をテクスチャとして取り込み、Unity の頂点シェーダにハイトマップとして扱わせることにします。試してみたところこの方法 であれば分割せずに表示することができました。

Unity に取り込むためにエクスポートする

DEM のベクタデータをラスタ化します。ラスタ化する際には縦横のピクセル数を決め なければいけません。国土地理院の基盤地図情報の仕様書を見ると、1 メッシュが 225 × 150 のデータ点で構成され、10 × 10 のファイルがひとつの Zip ファイルにまとめられ ています。今回の例ではそれを 2 × 2 並べているので、縦横のデータ点の数は、4500 × 3000 となります。

ッールボックスの「ラスタ化」を用いて DEM レイヤを GeoTIFF に変換します。「ラ スタ化 (ベクタのラスタ化)」をツールボックスから探しダブルクリックして実行します。 パラメータとして「入力レイヤ」には DEM のベクタレイヤを、画素値とするフィールド を指定する「Field to use for a burn-in value」には標高値のフィールド名を、出力する ラスタサイズの単位を指定する「Output raster size units」には、ピクセルを設定しま す。「Width」「Height」には先ほど計算した 4500 と 3000 を設定します。「出力領域」に は「…」をクリックして「レイヤー/キャンパス範囲を利用する」を選び DEM レイヤを 選択します。「出力データ型」は Float32 を設定し「ラスタ化」には「…」から保存する ファイル名を設定します。そして「バックグラウンドで実行」をします。

Q =JJ\$K(K\$\$90=JJ\$K)	? ×
パラメーター ログ	– – ×
入力レイヤ	
" DemDsmHeight [EPSG:4326]	
選択した地物のみ	
Field to use for a burn-in value [optional]	
1.2 dem	→ プロセッシングツールボックス
A fixed value to burn [optional]	🍬 🕐 🕀 🗞
0.000000	Search***
Output raster size units	「「変換(形式変換)
ピクセル	
Width/Horizontal resolution	
4500,00000	《 2億の設定
Height/Vertical resolution	フィールドの基本統計
8000.000000	
出力領域 (xmin, xmax, ymin, ymax)	
[139.6250277777777,139.874972222222,35.583861110776664,35.749972215223394 [EPSG:4326]	····································
出力バンドに指定されたnodata値を割り当てる [optional]	> Q 7r(1/2-1/
0.000000	🖾 💽 > Q ベクタージオメトリ
▼ 高度なパラメータ	> Q ベウター作成
追加の作成パラメータ [optional]	> Q ベウター選択
プロファイル 既定	▼ ○ < < < > < < < < < < < < < < < < < < <
	> Q べりタ作成
	> Q ラスタツール
🐵 🚥 検証 ヘルブ	> Q ラスタ地形解析
出力データ型	> Q ラスタ分析
Proat32	▼ > Q レイヤツール
Invert rasterization	
5入外化	> Raster miscellaneous
D:/data/GISDATA SAMPLE/dem.tif	
Cpen output file after running algorithm	↓ べの変換
GDAL/OGR コンソールコール	② 形式変換
gdal_rasterize -i layer_name -a dem -ts 4500.0 3000.0 -te 139.6250277777777 35.583361110776664 139.87497222222 3 nath to data file C://ligars/chore/AppLata/local/Term/processing 11b09041af804433871ecb115c587a53/544348231bis	35.749972215223394 -ot Float32 -of GTiff
	> 529抽出
	> フノダ役形 > ラスタ分析
	> ラスク変換
	> W GRASS
	27/17b
Run as Batch Process	ラウンドで実行 Close Help Help

▲図 6.36 DEM のラスタ化

同様に建築物の外周線データレイヤもラスタ化します。「ラスタ化(ベクタのラスタ 化)」を実行します。パラメータとして「入力レイヤ」には建築物の外周線データのベクタ レイヤを、画素値とするフィールドを指定する「Field to use for a burn-in value」には建 築物の高さのフィールド名を、出力するラスタサイズの単位を指定する「Output raster size units」には、ピクセルを設定します。「Width」「Height」には 4500 と 3000 を設定 します。「出力領域」には「…」をクリックして「レイヤー/キャンパス範囲を利用する」 を選び、先ほどラスタ化した DEM レイヤと合わせるために DEM レイヤを選択します。 「出力データ型」は Float32 を設定し「ラスタ化」には「…」から保存するファイル名を 設定します。そして「バックグラウンドで実行」をします。

Q =JJ\$K(K\$\$\$0=JJ\$K)	?	×
パラメーター ログ		
C> BuildingWithHeight [EPSQ:4326]	•••	2
選択した地物のみ		
Field to use for a burn-in value [optional]		
1.2 height		•
A fixed value to burn [optional]		
0.000000	×	+
Output raster size units		
ピクセル		•
Width/Horizontal resolution		
4500.000000	×	-
Height/Vertical resolution		
3000.000000	×	-
出力領域 (xmin, xmax, ymin, ymax)	_	
139.6250277777777,139.874972222222,35.583361110776664,35.749972215223894 [EPSG:4326]		•••
出力パンドに指定されたnodata値を割り当てる [optional]		
0.00000	×	\$
▼ 高度なパラメータ		
追加の作成パラメータ [optional] コロコーイル 「開業		_
70791W KAE		·
名称值		
名称 值		
名称		
名称 值		
名称 値 一 一 一 検証 ハルブ Hカデー 本型		
名称 値 一 御 一 検証 ヘルプ 上 カレプ Float32		•
名称 値 ● ● 検証 ヘルプ 出力データ型 Float32 Pre-initialize the output image with value [optional]		•
名称 値 回 検証 ヘルプ 出力データ型 Float32 Pre-initialize the output image with value [optional] 0.000000	•	•
名称 信 一 使証 本 が が が が が が が が で 、 な に 、 、 の し の の の の の の の の の の の の の		•
名称 値 ● ● 検証 ヘルノフ 出力デーク型 Float82 Pre-initialize the output image with value [optional] 0.000000 □ Invert rasterization 52.5%比 D. Cdsta / GEDDATA SAMPLE Availables tif	@	•
名称 値 ● ● 検証 ヘルプ 出力デー処型 Float32 Pre-initalize the output image with value [optional] 0.00000 □ Invert rasterization 3.2が比 Dr./data/GISDATA_SAMPLE/building tit	•	•
名称 値 ● ● 検証 ヘルプ 出力デー処型 Flat2 Pre-initialize the output image with value [optional] 0.00000 □ Invert rasterization 3.2.%比 D:/dat/GBDATA_SAMPLE/building tif ① Open output file after running algorithm GDAL/OGR コンソールール	8	•
名称 値 単正 ヘルプ 出力データ型 Float32 Pre-initialize the output image with value [optional] 000000 Invert rasterization スス化 P/data/GISDATA_SAMPLE/building.tif Open output file after running algorithm GDAL/OGR コンソールコール etal rasterize - I Javer name - a beieht -ts 4500.0 3000.0 -te 138.825027777777 85.583861110776664 138.87497222222 35.749972215228384 -ot Float5	@ []	
名称 値 単語 ヘルプ 出力データ型 Float32 Pre-initialize the output image with value [optional] 0.000000 Invert rasterization えスタ化 D/data/GISDATA_SAMPLE/building.tif Open output file after running algorithm GDAL/OGR コンソールコール gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.8250277777777 85.833811110776864 139.87497222222 35.74997221522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.8250277777777 85.833811110776864 139.87497222222 35.749972215223394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.8250277777777 85.83381110776864 139.87497222222 35.74997221522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.82502777777777 85.83381110776864 139.87497222222 35.74997221522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.825027777777777 85.83381110776864 139.87497222222 35.74997221522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.82502777777777 85.83381110776864 139.87497222222 35.74997221522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.82502777777777 95.583831110776864 139.87497222222 35.74997221522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.82502777777777 95.583831110776864 139.87497222222 35.74997221522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.82502777777777 95.583831110776864 139.87497222222 35.7499721522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.8250277777777 95.583831110776864 139.87497222222 35.7497221522394 - ot Float gtdaf rasterize - 1 layer, name - a height + 1s 4500.0 9000.0 - te 139.82502777777777 95.583831110776864 139.87497222222 35.7497221522394 - ot Float gtdaf rasterize - 1 gtdaf rast	≪ (32 −of	•
名称 値 単一 検証 ヘルブ 出力データ型 Flaa32 Pre-initalize the output image with value [optional] 0.000000 Invert rasterization 3スタ化 D/data/GEDATA_SAMPLE/building tif Open output file after running algorithm GDAL/OGR コンノールコール edal rasterize -1 layer name -a height -ts 4500.0 3000.0 -te 139.8250277777777 85583861110776664 139.874972222222 85.749972215223894 -ot Float: GTiff path to data lie C./Users/oho-s/AppData/Local/Temp/processing_11b990d1ef604d33871ecb115c587e53/9d326aa3164e4df6ac08ec7855bda00c/ OUTPUT tif	32 -of	
名称 値 単一 検証 ヘルプ 出力データ型 Fbat32 Pre-intialize the output image with value [optional] 0.000000 Invert rasterization 3スタ化 D/data/GEDATA_SAMPLE/building tif Open output file after running algorithm GDAL/OGR エンジールニール edal rasterize -1 layer name -a height -ts 4500.0 3000.0 -te 139.6250277777777 85.583361110776664 139.874972222222 35.749972215223394 -ot Float: GTiff path to data file C./Users/oho-s/AppData/Local/Temp/processing_11b990d1et604d33871ecb115c587e53/9d326aa3164e4df6ac08ec7655bda00c/ OUTPUT tif	32 -of	•
名称 値 単一 検証 ヘルブ 出力データ型 Float32 Pre=intialize the output image with value [optiona] 0.000000 ① Invert rasterization 32.5/1と D/data/GISDATA_SAMPLE/building tif ② Open output file after running algorithm GDAL/OGR エンジールニール edal rasterize -1 layer name -a height -ts 4500.0 3000.0 -te 139.62502777777777 85583361110776664 139.874972222222 35.749972215223394 -ot Float GTirft path to data file C/Users/oho-s/AppData/Local/Temp/processing_11b990d1et604d33871ecb115c587e53/9d328aa3164e4df6ac08ec7655bda000c/ OUTPUT.tif	32 -of	•
名称 値 単一 検証 ヘルブ 出力データ型 Fbat32 Pre-initialize the output image with value [optiona] 0.000000 □ Invert rasterization 32.3/1と D/data/GISDATA_SAMPLE/building tif ② Open output file after running algorithm GDAL/OGR エンソールレール edal rasterize -1 layer_name -a height -ts 4500.0 3000.0 -te 139.62502777777777 85.583361110776664 139.874972222222 35.749972215223394 -ot Float: GTirft path to data file C:/Users/oho-s/AppData/Local/Temp/processing_11b990d1et604d33871ecb115c587e53/9d328aa3164e4df6ac08ec76558bda0Uc/ OUTPUT.tif	32 -of	•
名称 値 単力データ型 Fbat32 Pre-initalize the output image with value [optional] 0.000000 0.000000 0.000000 0.000000 0.00000 0	۲ (132 – of (132 – of	 ▼ ■ ■
名称 使 ● 検証 ヘルブ 出力データ型 Float32 Fre=nitalize the output image with value [optional] 0.000000 Invert rasterization 3スタ化 D/data/GISDATA_SAMPLE/building tif Open output file after running algorithm GDAL/OGR コンソールコール gdal_rasterize -1 layse_rname -a height -ts 45000 3000.0 -te 139.8250277777777 35583361110776664 139.874972222222 35.749972215223394 -ot Float GTiff path / odday file C./Users/oho-s/AppData/Local/Temp/processing_11b990d1ef604d33871ecb115c587e53/9d328aa3164e4df8ac08ec7655bda00c/ OUTPUT.tif		•

▲図 6.37 建築物の外周線データのラスタ化

次に、ふたつのラスタレイヤの加算を行います。メニューの「ラスタ」-「ラスタ計算 機」を選択します。



▲図 6.38 メニュー:ラスタ計算機

ラスタ演算式に行いたい演算の式を作成します。Dem のラスタレイヤ名を「ラスタバ ンド」フィールドで選びダブルクリックすると演算式に追加されます。演算子から「+」 を選びクリックして演算式に追加し、建築物の外周線データのラスタレイヤ名をダブルク リックして「"BuildingRaster@1" + "DemRaster@1"」のように演算式を完成させてくだ さい。「ラスタ演算式」のフィールドの下部に「式は正しいです」という表示があれば、正 しく解釈されています。なお、ラスタバンドフィールドのレイヤ名の後の「@1」という表 記はバンド1という意味です。バンドは RGB 画像でいう R や B や G のカラーチャネル のような意味で、今回使っているラスタデータはシングルバンドのデータですが、バンド が複数あるマルチバンドのラスタデータでは@以降の数字が異なる複数のレイヤ名が表示 されます。

「出力レイヤ」で「…」から保存するファイルを選択し、「出力形式」を GeoTIFF に設定します。「選択レイヤの領域」をクリックすれば自動的に計算する領域を利用するレイヤの領域に合わせてくれます。「OK」で実行します。

Q ラスタ−計算機									?	×
ラスタバンド				ラスタ	レイヤー					
N035E 100 AV	DCM01			出力に	শপ 🚺	D:¥data¥GISDA	TA_SAMP	LE¥newDSM	63	
DemRaster@1				出力刑	沅	GeoTIFF			_	-
				選択	レイヤの領域					
				X最小	値 139.62503	\$	X最大値	139.87497		٢
				Y最小	値 35.58336	\$	Y最大值	35.74997		-
				カラム	4500	\$	行	3000		-
				出力の	RS	EPSG:4326 - V	/GS 84		•	
				☑ 結	果をプロジェクト(ご追加する				
▼ 演算子										
•	*	平方根	COS	sin	tan	log 10		(
-	/	· ·	acos	asin	atan	In)		
<	>	=	i=	<=	>=	AND		OR		
ラスタ演算式										
"BuildingRas	ter@1" + "	DemRaster@1"								
式は正しいです						_				

▲図 6.39 ラスタ計算機ダイアログ

最後に DEM と建築物の高さデータを合わせたラスタの数値の型を UInt16 に変換して
保存します。符号なし整数にしないと Unity ではテクスチャとして扱いにくいためです。 そのためには、最小値を計算しておかなければなりません。

メニューから「ラスタ」-「その他」-「ラスター情報」を選びます。「入力レイヤ」に対 象のレイヤを、「各バンドの実際の最小/最大値の強制計算」と「画像統計量を読み込んで 表示する(必要に応じて計算を強制する)」にチェックを入れ、「バックグラウンドで実行」 します。

Q , ⋽スター情報		?	×
パラメーター ログ			
入力レイヤ			
PrewDSM [EPSG:4326]		-	•••
✓ 各バンドの実際の最小/最大値の強制計算			
── 画像統計量を読み込んで表示する(必要に応じて計算を強制する)			
CP情報を出力しない			
□ メタデータ情報を出力しない			
レイヤ情報			
[一時ファイルへの保存]			•••
GDAL/OGR コンソールコール			
gdalinfo -mm -stats D:¥data¥GISDATA_SAMPLE¥newDSM.tif			
	0%	キャン	セル
Run as Batch Process	バックグラウンドで実行 Close	He	elp

▲図 6.40 ラスタ情報ダイアログ

計算が終わり「ログ」タブに結果が出力されます。これを見ると最小値は-35.05 と分かります。

Q ⋽スター情報		?	×
パラメーター ログ			
Upper Right (139.8749700, 35.7499700) (139d52'29.89"E, 35d44'59.89"N)			^
Lower Right (139.8749700, 35.5833600) (139d52'29.89"E, 35d35' 0.10"N)			
Center (139.7500000, 35.66666650) (139d45' 0.00"E, 35d39'59.99"N)			
Band 1 Block=4500x1 Type=Float32, ColorInterp=Gray			
Min=-35.050 Max=224.270 Computed Min/Max=-35.050,224.270			
Minimum=-35.050, Maximum=224.270, Mean=17.916, StdDev=17.578			
NoData Value=-3.4028234663852886e+38			
Metadata:			
STATISTICS_MAXIMUM=224.26998901367			
STATISTIC5_MEAN=17.916132822413			~
	0%	キャンセノ	l.
Run as Batch Process***	バックグラウンドで実行 Close	Help	

▲図 6.41 ラスタ情報結果

なお、ベクタデータ用にも同様な統計量計算ツールが用意されています。活用する場面 があるかと思います。

次に UInt16 に収まるようにラスタ計算をします。得られた最小値をもとにして最小値

分を足すことにより0以上の値域に調整します。また、UInt16では0~65535までの数 値を扱えます。富士山の標高が3776mなので、日本国内のデータを扱うならば10倍して もUInt16の値範囲に収まります。10倍することで小数点以下1桁の10cmオーダーまで の精度を持たせることができます。ラスタ計算機を用いてこの計算を行います。計算式は ("newDSM@1" + 35.05)*10となります。

Q ∋スター計算機					? ×
ラスタバンド	- 57	.ቃレイヤ			
N035E139_AVE_DSM@1	出7	うレイヤ	ta¥GISDATA_SAMF	PLE¥newDSMranged	a 🛄
BuildingRaster@1 DemRaster@1		り形式 🗧	GeoTIFF		-
newDSM@1	選	択レイヤの領域			
	׿	小値 139.6250	3 🗘 Xi	最大値 139.87497	\$
	YÆ	小値 35.58336	÷ Yi	最大値 35.74997	
	カラ	La 4500	‡ 13	3000	÷
	出力	ካርRS	EPSG:4326 - WGS	84	-
		結果をプロジェクト	トに追加する		
▼ 演算子		_			
+ * 平方根	cos sin	tan	log 10	(
- / ^	acos asin	atan	In)	
< > =	!= <=	>=	AND	OR	
ラスタ演算式					
("newDSM@1" + 35.05)*10					
式は正しいです			_	1	
			OK	Cancel	Help

▲図 6.42 ラスタ計算機での値域の調整

作成されたラスタデータはまだ Float32 形式なので、これを UInt16 形式の GeoTIFF に変換して保存します。

メニューから「ラスタ」-「変換」-「変換(形式変換)」を実行します。「入力レイヤ」に 値域を調整したラスタレイヤ名を、「出力データ型」に UInt16 を、「変換された」に「…」 から保存先ファイル名を指定して入力して、「バックグラウンドで実行」します。

Q 変換(形式変換)		?	×
パラメーター ロガ			
λπυ/ヤ			
www.commonwedlepsc.used		-	•••
出力ファイルの投影を上書きする [optional]			
		•	
出力バンドに指定されたnodata値を割り当てる [optional]			
0.000000		e	1
□ このファイルのサブデータセットを個別出力ファイルにコピーする			
▼ 高度なパラメータ			
返加の作成パラメータ [optional] ゴロフェイル 歴史			-
/U//1// MAE			•
名称值			
金 ● 検証 ヘルプ			
出力データ型			_
Unt 16			•
363RC41/C		- 1	
GDAL/OGR コンソールコール			
gdal_translate -ot UInt16 -of GTiff D¥data¥GISDATA_SAMPLE¥newDSMranged.tif C:/Users/oho-s/AppData/Local/Temp/			
processing_11b990d1ef604d33871ecb115c587e53/df843ac6f143451c95a24a93f8c0e948/OUTPUT.tif			
V			
	0%	キャ	d211
	-	- n n 2	
Run as Batch Process***	हींने Close	He	elp

▲図 6.43 UInt16 型で保存

これでハイトマップ用の画像の処理とエクスポートができました。

書き出した画像の処理

16 bit TIFF でもまだ Unity では扱いにくいので、ImageJ を利用してカラーマップと 法線マップを計算するための 8 bit 画像を作成します。ImageJ は科学研究での画像解析 などに広く利用される画像処理ソフトウェアです。今回は 16 bit TIFF を読み込めるこ とや正確な処理が可能であることなどから使用します。ImageJ の Web ページ*¹⁴からイ ンストーラをダウンロードしてインストールしてください。

メニューの「File」-「Open」から先ほど UInt16 で保存した GeoTIFF 画像を開き ます。8 bit 画像にするまえに明るさ補正をします。メニューの「Image」-「Adjust」-「Brightness/Contrast」を選択します。

^{*14} ImageJ の Web ページ:https://imagej.nih.gov/ij/

			Туре	•	
	Brightness/Contrast	Ctrl+Shift+C	Adjust	•	
🛓 newDSMUInt	Window/Level		Show Info	Ctrl+I	
4500x3000 pixels	Color Balance		Properties	Ctrl+Shift+P	
	Threshold	Ctrl+Shift+T	Color	•	
	Color Threshold		Stacks	•	
at the second of	Size		Hyperstacks	•	
	Canvas Size		Crop	Ctrl+Shift+X	A Receive
	Line Width		Duplicate	Ctrl+Shift+D	1 Contraction
	Coordinates		Rename		
1.	Auto Threshold		Scale	Ctrl+E	5 L-11-5
a	Auto Local Threshold		Transform	•	
	and a straight from the state		Zoom	•	
			Overlay	•	
			Lookup Tables	,	AND R.

▲図 6.44 ImageJ で明るさ補正をする

明るさ補正のダイアログで、Minimum をスライドバーの最小に Maximum をスライド バーの最大にします。



▲図 6.45 明るさ補正ダイアログ

そして、メニューの「Image」-「Type」-「8-bit」を選択し、色深度を8 bit に変換します。



▲図 6.46 色深度を 8 bit に変換

最後に別名で保存してください。この画像ファイルがカラーマップとなります。

次に法線マップを作成します。法線マップは GIMP のプラグインを利用します。

まず GIMP の Web ページ*¹⁵からインストーラをダウンロードしてインストールして ください。

そして、法線マップを作成するプラグインを Web サイト*¹⁶からダウンロードしてイン ストールします。ダウンロードした Zip アーカイブの readme.txt にあるように、アーカ イブ内のファイルを配置してください。

GIMP で画像を開きます。メニューの「ファイル」-「開く/インポート」や、直接 GIMP のウインドウに画像ファイルをドラッグ&ドロップすることで開きます。

画像を開いたら、メニューの「画像」-「モード」-「RGB」を選択して、グレースケー ル画像から RGB 画像に変換します。



▲図 6.47 RGB 画像に変換

そして、メニューの「フィルター」-「マップ」-「Normalmap…」を選択します。

^{*&}lt;sup>15</sup> GIMP Web ページ:https://www.gimp.org/

^{*&}lt;sup>16</sup> GIMP 法線マッププラグインの URL: https://code.google.com/archive/p/gimp-normalmap/

1-11(T)	フィルター(<u>R</u>)	ウィンドウ(W)	ヘルプ(日)		
1500,	 クイルター カイルター フィルター すべての 	の再適用 の再表示 フィルターのリセッ	Ctrl+ Shift+Ctrl+ / h(<u>F</u>)	+F +F +F	
	(新)(<u>B)</u> (新)調形(D) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1))) 出(I)) (功果(<u>A</u>)			
	マップ(M 下塗り(E ウェブ(W	D 3))		 Normalmap オフシェクトにマップ(Q) シームレス化(M) 	a
	アニメーS ロゴ効果 Duthon	۲۹۷() (L) د			

▲図 6.48 法線マッププラグインの実行

ダイアログでは、Filter に試しに「Sobel 3x3」を選んで実行してみます。

🥶 Normalmap				×
Preview	Filter:	Sobel 3x3	~	Options
	Minimum Z:	0.00000	*	Invert X
	Scale:	1.00000	•	Swap RGB
	Height source:	Average RGB	\sim	
	Alpha channel:	Unchanged	\sim	
	Conversion:	None	\sim	
3D Preview	DU/DV map:	None	\sim	
(空) ~	Contrast:	0.00000	A V	
		キャンセル	(<u>C</u>)	OK(<u>O</u>)

▲図 6.49 法線マッププラグインのダイアログ

青い法線マップ画像が作成されました。これも別名で TIFF などにエクスポートしま す。この画像が、法線マップとなります。

最後に元の 16 bit の GeoTIFF 画像を加工しましょう。このままでは 16 bit の値域 をそのまま Unity に取り込むことができないので工夫します。まず、ファイルのプロパ ティでサイズを確認します。例で用いている画像の場合は 27,024,402 バイトでした。画 像データ自体のデータ量は 4500 × 3000 × 2 で 27,000,000 バイトなので、24,402 バイト がヘッダなどの画像データ以外のデータとなります。そこで、この画像ファイルをバイナ リエディタで読み込み、先頭から 24,402 バイトまでを削除して、別名で保存します。筆 者はバイナリエディタとして、Stirling^{*17}を使っていますが、同等の操作が可能であれば 何を使っても構いません。

次に、リスト 6.1 を用いて 16 bit で格納されていたデータを 24 bit にして RGB のデー タ構造に合わせます。リスト 6.1 は Visual Studio などで適切にコンパイルして実行して ください。また、ファイル名などは適宜変更してください。

▼リスト 6.1 16toRGB.c

```
#include <stdio.h>
int main() {
        unsigned char buf[3];
        FILE *fp;
        FILE *fp2;
        // 入力ファイル
        fopen_s(&fp, "dsmraster_uint16.raw", "rb");
        // 出力ファイル
       fopen_s(&fp2, "dsmraster_uint16_rgb.raw", "wb");
       for (int i = 0; i < 4500 * 3000; ++i) {
                fread(buf, 2, 1, fp);
                buf[2] = 0x00;
               fwrite(buf, 3, 1, fp2);
        7
        fclose(fp);
       fclose(fp2);
        return 0;
}
```

最後に、GIMP でこのバイナリファイルを RAW 画像データとして読み込み、TIFF に 書き出します。GIMP のメニューから「ファイル」-「開く/インポート」を選択します。 対象のバイナリファイルを選択したら、ファイル形式の選択で「Raw 画像データ」を選択 して開きます。

^{*17} Stirling ダウンロードページ:https://www.vector.co.jp/soft/win95/util/se079072.html

· 新行(P)	名前 [m B D 2	• 9 1%	最終変更日 ^	プレビュー(<u>E</u>)
の長い思いたついてい	E BLD3		2018/04/02	-
D RADRYS/71/P	BLD4		2018/04/02	5 m 35
= oho-s	ED DEM		2018/04/03	March 1
Wedness (5)		54.0 MB	2018/04/06	Sector States
- D-DIL 5(77/D)	a dem tif	54.0 MB	2018/04/06	
DVD RW KS/T/E) T	a demUINT16.tif	27.0 MB	2016/04/05	newDSMrabRAW.til
USB F5-17 (G)	newDSM.tif	54.0 MB	2018/04/06	40.5 MB
PS4 (Ot)	newDSM8bit.tif	13.5 MB	21:43	4500 × 1000 E7 E/- RGR, 18001-17-
Documents	newDSM8bitgray.tif	27.1 MB	2018/04/06	
Pictures	newDSMnormal.tif	40.5 MB	22:05	
	newDSMranged.bif	54.0 M8	2018/04/06	
	newDSMrgbRAW.tif	40.5 MB	22:32	
	newDSMUInt16.tif	27.0 MB	2018/04/06	
	newDSMUInt16RAW.tif	27.0 MB	22:13 V	
+ -		すべての画像		
7テイル形式の選択(① (現ま	生の設定: Raw画像データ)			
ファイル形式 PUF だい				拡張子 pat
Photoshop 画像				psd
PNG 画像				png
PNM 画像				pnm,ppm,pgm,pbm
PostScript ドキュメント				ps
Raw画像データ				data
Silicon Graphics IRIS 黃像				sgi,rgb,rgba,bw,icon
SUN-raster ファイル直像				im1, im8, im24, im32, rs, ras

▲図 6.50 GIMP で Raw 画像データとして開く

「Raw データから画像読み込み」のダイアログで、画像の種類に「RGB」幅を「4500」 高さを「3000」として開きます。

■ Rawデータから画像読み込み	0
ER REPERT. SCA	
(1) 10 10 10 10 10 10 10 10 10 10 10 10 10	
高金 高の登場(D) R58 オフなり(D) - 何のの	
新会 「 新会の理想()」 A58 オフセントロン 国政 日本 日本	
高齢 単本の理想(D): 5.55 オフセント(D): 単位(D): 高を(D):	0 4501 3000 0
高令 単本の世話(1): PG6 オフをわし、 単位の 単位の 一 応告 ・ パンマト し、 ・ の し、 の の の の の の の の の の の の の	
また	
高かり増加(1): RGB オンセット(1): RGB 花り(1): RGB 花り(1): RGB 花り(1): RGB だいっかで増加(1): RGB (直常) オンセット(2): RGB た日 た日 た日 た日 た日 た日 た日 た日 た日 た日	

▲図 6.51 Raw 画像データとして読み込むパラメータ設定

画像が表示されたのを確認して RGB の TIFF としてエクスポートします。この画像を ハイトマップとして使用します。

6.10 Unity から使う

最初にベースとするための 100 × 100 分割した四角い平面ポリゴンモデルを作成しま す。今回は使い慣れている Metasequoia 4 Standard^{*18}を使いました。

同等の事ができれば使うソフトウェアは何でも構いません。作成したポリゴンモデルは OBJ などの Unity に読み込める形式で保存してください。



▲図 6.52 Metasequoia でのモデル作成

さて、最後の工程です。Unity に取り込んで表示してみましょう。Unity のバージョン は Unity 2017.4.1f1 を用いました。最近のバージョン (2017 系以降) であれば問題ない と思います。新規プロジェクトを作成します。

^{*&}lt;sup>18</sup> Metasequoia のページ:http://www.metaseq.net/jp/

最初にハイトマップ、カラーマップ、法線マップとポリゴンモデルを読み込みます。 Project ペインにドラッグ&ドロップするなどしてプロジェクトにインポートしてくだ さい。テクスチャの設定は図 6.53 のようにしてください。法線マップのみ、「Texture Type」を Normal Map にしておきます。



▲図 6.53 テクスチャの設定

次に、適当なシェーダーファイルを新規作成して、リスト 6.2 のシェーダを入力します。

```
▼リスト 6.2 Terrain.shader
```

```
Shader "Terrain" {
       Properties{
                _Tess("Tessellation", Range(1,128)) = 4
                 _MainTex("Base (RGB)", 2D) = "white" {}
        _DispTex("Disp Texture", 2D) = "gray" {}
            _NormalMap("Normalmap", 2D) = "bump" {}
            _Displacement("Displacement", Range(0, 1.0)) = 0.3
                _Color("Color", color) = (1,1,1,0)
                _SpecColor("Spec color", color) = (0.5,0.5,0.5,0.5)
        7
        SubShader{
                Tags{ "RenderType" = "Opaque" }
                LOD 300
                CGPROGRAM
#pragma surface surf BlinnPhong addshadow fullforwardshadows
        vertex:disp tessellate:tessFixed nolightmap
// ↑紙面の都合上二行に分けていますが一行です。
#pragma target 4.6
                struct appdata {
                float4 vertex : POSITION;
                float4 tangent : TANGENT;
                float3 normal : NORMAL;
                float2 texcoord : TEXCOORDO;
        };
        float _Tess;
        float4 tessFixed()
        {
            return _Tess;
        7
        sampler2D _DispTex;
        float _Displacement;
        void disp(inout appdata v)
        Ł
            float d = (tex2Dlod(_DispTex, float4(v.texcoord.xy,0,0)).r
                        + tex2Dlod(_DispTex, float4(v.texcoord.xy, 0, 0)).g * 256)
                        * _Displacement;
            v.vertex.xyz += v.normal * d;
        }
        struct Input {
            float2 uv_MainTex;
            float2 uv_NormalMap;
        };
        sampler2D _MainTex;
        sampler2D _NormalMap;
        fixed4 _Color;
        void surf(Input IN, inout SurfaceOutput o) {
            half4 c = tex2D(_MainTex, IN.uv_MainTex) * _Color;
            o.Albedo = c.rgb;
            o.Specular = 0.2;
            o.Gloss = 1.0;
            o.Normal = UnpackNormal(tex2D(_NormalMap, IN.uv_NormalMap));
        7
        ENDCG
        }
        FallBack "Diffuse"
}
```

シェーダの処理としては、固定値のテッセレーションでポリゴンを分割し、頂点シェー

ダでハイトマップ分を法線方向に押し出し、フラグメントシェーダでカラーマップと法線 マップをもとに描画しています。頂点シェーダでの計算は、ハイトマップのR値を下位8 ビットに、G値を上位8ビットに復元する処理となっています。

そして、マテリアルを新規作成して Shader をリスト 6.2 のシェーダに設定し、用意した 3 枚のテクスチャと各パラメータをそれぞれ設定します。Tessellation の値は 32 くらいに設定します。正確な高さが必要な場合は計算する必要がありますが、Displacement の値は様子をみながら調整してください。

	🕑 Collab 🔻	🛆 Account	•	Layers 🔻	Layout 👻
Project Create + Q V Creash	4 9	 Inspector Terrain Shader 	Terrain	clusion	≞ +≡ ₪ &,
default ● defaultMat ⊞ default ◎ newDSM8bit		Tessellation Base (RGB)			32
newDSMnormal		Tiling Offset	× 1 × 0	Y 1 Y 0	Select
Terrain S TerrainShader		Disp Texture			
		Tiling Offset	X 1 X 0	Y 1 Y 0	Select
		Normalmap			
		Tiling Offset	X 1 X 0	Y 1 Y 0	Select
		Displacement			0.3
		Spec color			
		Render Queu			¢ 2000
		Enable GPU I Double Sided	nstancing I Global I) Ilumination	

▲図 6.54 マテリアルの設定

最後に Hierarchy にポリゴンモデルを追加し、テクスチャのアスペクト比に合わせて Scale を (45,1,30) などに設定し、先ほど用意したマテリアルを設定します。



▲図 6.55 メッシュオブジェクトの設定



いかがでしょうか、図 6.56 のように GIS から作成した都市が表示されたと思います。

▲図 6.56 結果の表示

6.11 おわりに

今回はデータを画像に落とし込み Unity と連携させましたが、ベクタデータのまま Unity の 3D オブジェクトに変換できると、より表示精度が高くなります。また、Mesh を加工することでゲームなどに使いやすくできると思います。そのためには GIS データ を自分の使いたい形式にする独自コンバータの作成などが必要になってきます。さらに、 より広い範囲を扱いたい場合はこの巨大なデータを扱うための空間分割をしなければなり ません。

機会があれば次はそういった発展的な内容に挑戦したいと思います。

地理情報は背景となる知識やノウハウが専門的であり、またそもそもあまり情報がない こともあって、かなりとっつきにくい分野ではあると思います。しかし、このように実際 に操作しつつ扱ってみると、意外と何とかなりそうと感じられた方もいるかと思います。

今回は目標として、Unity に実際の都市を取り込むというゴールを設定してその流れを 説明しましたが、GIS はそのほかにもさまざまな処理や分析ができます。災害対策や商圏 分析、環境保護、資源探査、そしてもちろんゲーム、さまざまな分野で有効な仕組みです。 本稿で少しでも興味を持ち面白いと思っていただけるのであれば、著者として望外の喜び です。

Suguru Oho / @ohomagic

著者

第1章 Daisuke MAKIUCHI / @makki_d

眼鏡っ娘が好きです。

第2章・第3章 Kinuko MIZUSAWA

ふと思ったのですが、最近やっとターミナルとお近づきになれてきた気がします。 個人的にはもうちょっとデレてくれてもいいと思います。

第4章やまだ

花粉症だと認めたくない。

第5章 Shunsuke Ito / @fgshun

RimWorld プレイ中。さて明日はどこに墜落しようかな。

第6章 Suguru Oho / @ohomagic

最近の楽しみは4歳の息子とGT SPORT をプレイすることです。

表紙 たなか

コメント書くのは苦手です。

裏表紙 いしい

久々に仕事で絵描いた

ちびキャラ あべ

最近ガチャの引きが悪い

ダウンロードカード こにし

太眉はいいぞ

ダウンロードカード あきやま

久しぶりに女の子描きました

編集 岡本和樹 / @kakkun61

QoL 爆上げとの噂のドラム式洗濯乾燥機を買った。まだ宅配来てない。

KLab Tech Book Vol. 2

2018年4月22日 技術書典4電子版(2.1)著者 KLab技術書サークル編集 岡本和樹発行所 KLab技術書サークル

(C) 2018 KLab 技術書サークル

123