Minecraft Overviewer 導入/設定論@自宅小規模鯖

非ルー記

目次

1 はじめに

Minecraft Overviewer 導入に際しての日本語まとめが少なすぎるんだよこの世の中は！俺が作るぞ！

※本資料は minecraft v1.19 までしか使用できません。v1.20 で使えなくなったのでご承知ください。

2 Minecraft Overviewer is 何

すげえ簡単に言うと **稼働してるマイクラのマップをブラウザで表示する** ツールです。リアルタイム性はなく、ゲーム内で表示されるわけでもありませんが、代わりにかなり細かく表示されますし、MOD ではないのでバニラでも使用可能です。図 1 みたいな感じで出ます。

図 1: Overviewer 出力例

2.1 何ができて何ができないの？

2.1.1 できること

- ディメンション毎のマップ出力[1)]
- 看板等の設置物の出力
- 特定条件による色付け[2)]

2.1.2 (筆者がやりたかったけど) できなかったこと

- MOD ブロックの出力
- 洞窟系の出力[3)]

2.1.3 その他

- CUI[4)]操作は必須です。win の GUI しか触ったことない人もこの期に CUI を触ってみよう！
- Java 版のみ動作確認しています。BE は……使えるのか誰か試してみてほしい……
- Ubuntu 系は筆者が詳しくないので本資料では記載しません。rhel にしよう！
- サーバ公開を目指した内容ではありますが、本筋ではないのでセキュリティについてはガン無視です。各自で調べて担保してください。
- **筆者の自由研究みたいなものなので、Overviewer 本家にこの本について問い合わせるのはおやめください。**

1) オーバーワールド、ネザー、エンドなど
2) プリセットの範囲で可能
3) できるんだけどかなり読みにくいものになる
4) Character-based User Interface。コマンドプロンプトとかの類。CLI(Command Line Interface) とも言う。

3 準備

導入前の確認事項と前提知識の確認です。

3.1 Overviewer の必要環境

動作に必要なもの：

- minecraft の world データ
- 確認用のブラウザ[5)]
- それなりのスペックのマシン[6)]

3.2 world データについて

大体の場合.minecraft の中にある、以下のようなファイル構成のディレクトリです。サーバだとまんま「world」というディレクトリになっていることが多いですが、シングルだと「saves」の中にあったりします。

```
==========================
<DIR>          advancements
<DIR>          data
<DIR>          datapacks
<DIR>          DIM-1
<DIR>          DIM1
<DIR>          entities
        11,519 icon.png
         2,081 level.dat
         2,082 level.dat_old
<DIR>          playerdata
<DIR>          poi
<DIR>          region
             3 session.lock
<DIR>          stats
==========================
```

5） 今の時代にネスケとか IE とか使ってなければ多分何でも ok。
6） それなりに重い。メモリよりもコアに負荷が掛かるタイプ。

4 Overviewer 導入〜実行 (Windows11)

windows はマジで導入が楽なので、OS 代出せてサーバ用 OS ではない点を潰せるスペックを用意できるなら win 一択だと思います。そう、ここは互換性の揺り籠……

4.1 インストール

4.1.1 本体以外のダウンロードとインストール

Overviewer 本体の前に、まずは必要な環境をダウンロードします。VC++二種のインストールは DL した exe を実行して、「はい」「OK」を連打しておけば問題なし。

Microsoft Visual C++ 2008 Service Pack 1 再頒布可能パッケージ

`https://www.microsoft.com/ja-jp/download/details.aspx?id=26368`

Microsoft Visual C++ 2010 Service Pack 1 再頒布可能パッケージ

`https://www.microsoft.com/ja-jp/download/details.aspx?id=26999`

リソースパック

minecraft(クライアント側) がインストールされたディレクトリの `.minecraft/versions/<バージョン>/<バージョン>.jar` を確保しておきます。どこかにコピーしておいてもよいし、Overviewer と同じマシンにあるなら場所をメモしておくでも ok。ファイル名の例としては「`1.19.2.jar`」など。

4.1.2 Overviewer インストール

Overviewer をダウンロードします。

Downloads - Minecraft Overviewer

`https://overviewer.org/downloads`

「Windows」の「64-bit」を選んでおけば ok。~~32bitOS はそもそも動かすのを諦めた方がよい~~
ダウンロードしたら解凍して好きな場所に置いてください。

4.2 config 作成

Overviewer を動作させるための設定ファイルが必要です。先ほど解凍して配置した Overviewer のディレクトリ内に「＜任意の名前＞.conf」というテキストファイル (コード 1) を作成してください。

ソースコード 1: config ファイル (win)

```
worlds["survival"] = "E:\game\minecraft\\1_18_2_Optifine\saves\world1"

renders["survivalday"] = {
    "world": "survival",
    "title": "Survival Daytime",
    "rendermode": smooth_lighting,
    "dimension": "overworld",
}


outputdir = "E:\game\minecraft\overviewer\mcmap"
texturepath = "E:\game\minecraft\overviewer\\1.19.2.jar"
```

worlds

Overviewer に読み込ませる world データの配置先を設定します。

world

どのワールドのマップを生成するのかを指定します。worlds の [] 内に指定した文字列と合わせておけば ok。

title 生成したマップのタイトルを指定します。この名前で表示されます。

rendermode

使うレンダリングのモードを指定します。光の表示のされ方とか生成速度とかが変わります。主なオプションは後述。

dimention

どのディメンションを生成するか指定します。`overworld`、`nether`、`end` などが使用可能。

renders

出力したいディメンションを設定します。上の例はオーバーワールドのものになります。

outputdir

生成したマップの出力先ディレクトリを設定します。沢山ファイル生成されるのでちゃんと専用のディレクトリ切るのが無難。

texturepath

4.1.1 のリソースパックを置いた場所を記載します。

その他注意事項

ファイルパスに「`\1`」を含む場合、「`\`」でエスケープする必要があります。

4.3 実行

1. 4.1.2 で Overviewer を配置したディレクトリに移動し、アドレスバーで「cmd」を入力しコマンドプロンプトを起動します (図 2)。

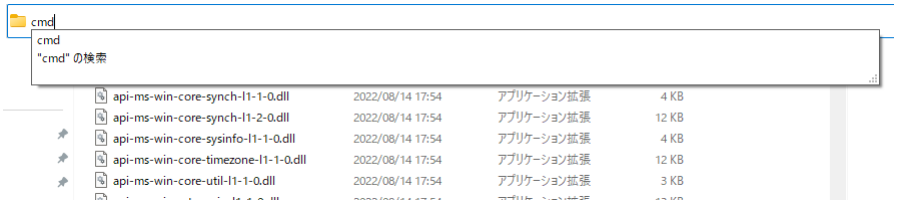

図 2: コマンドプロンプト起動

2. Overviewer が配置されているディレクトリでコマンドプロンプトが起動しますので、下記コマンドを実行します。config ファイルは 4.2 で作成したものとなります。

```
> overviewer.exe -c "＜config ファイルのパス＞"
```

3. 実行してエラーが出なければ、しばらくするとマップが生成されます！ 4.2 の **outputdir** で指定したパスの「**index.html**」を開き、マップが生成されたことを確認してください。

5 Overviewer 導入〜実行 (rhel9)

win の環境がどうしても用意できない人向け。インストールに際してソースからのビルドが必要なので、どうしても難易度は上がります。rhel7/CentOS7 なら yum でインストールできるので、終わった OS であるというリスクを抱えてでもそっちを選ぶ、というのも一応選択肢。

ソースからのビルドができると開発中のバージョンとかも使えるようになるので、そういう意味では出来た方が便利かも。

5.1 インストール

rhel のインストールと dnf による OS アップデートくらいまではできる前提で書いています。CentOS は死んだので rhel の無償利用とか fedora とかを使いましょう。

5.1.1 本体以外のダウンロードとインストール

Overviewer 本体の前に、まずは必要な環境をダウンロードします。

EPEL9

Overviewer を動かすために必要な python3 系パッケージをインストールするために必要なリポジトリ。以下コマンドにより導入します。

```
# subscription-manager repos --enable codeready-builder-for-rhel-9-$(arch)-rpms
# dnf install \
  https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```

python3-devel

Overviewer ビルドに必要なパッケージその 1。下記コマンドで導入します。

```
# dnf install python3-devel -y
```

python3-pillow

Overviewer ビルドに必要なパッケージその 2。下記コマンドで導入します。~~当初これに気付かなくて Pillow 直 DL して Imaging.h とか抜き出してた~~

```
# dnf install python3-pillow -y
```

gcc Overviewer ビルドに必要なパッケージその 3。下記コマンドで導入します。

```
# dnf install gcc -y
```

リソースパック

minecraft(クライアント側) がインストールされたディレクトリの .minecraft/versions/<バージョン>/<バージョン>.jar を確保しておきます。どこかにコピーしておいてもよいし、Overviewer と同じマシンにあるなら場所をメモしておくでも ok。ファイル名の例としては「1.19.2.jar」など。~~win のコピペ~~

5.1.2 Overviewer インストール

1. Overviewer のソースを DL します。git でもいいし、zip 落として unzip コマンドでも ok。DL 元 ⇒ https://github.com/overviewer/Minecraft-Overviewer
2. ダウンロードして展開したディレクトリ (setup.py がある場所) で以下コマンドを実行します。エラーがないことを祈ろう。

```
# dnf install git -y
# git clone https://github.com/python-pillow/Pillow.git
# PIL_INCLUDE_DIR="./Pillow/src/libImaging" python3 ./setup.py build
```

3. 上手くいったら以下のような感じでファイルが出揃うはずです。overviewer.py がいれば ok。

```
# ls
CONTRIBUTORS.rst  README.rst   contrib            overviewer.py     setup.cfg
COPYING.txt       build        contribManager.py  overviewer_core   setup.py
MANIFEST.in       build-tools  docs               sample_config.py  test
```

5.2 config 作成

Overviewer を動作させるための設定ファイルが必要です。5.1.2 で生成された Overviewer.py のあるディレクトリ内に「<任意の名前>.conf」というテキストファイル (コード 2) を作成してください。

ソースコード 2: config ファイル (rhel)

```
worlds["survival"] = "/root/ov_build/Minecraft-Overviewer/world"

renders["survivalday"] = {
    "world": "survival",
    "title": "Survival Daytime",
    "rendermode": smooth_lighting,
    "dimension": "overworld",
}

outputdir = "/root/ov_build/Minecraft-Overviewer/ov_map"
texturepath = "/root/ov_build/Minecraft-Overviewer/1.19.2.jar"
```

config の内容については 4.2 を参照。win と rhel で必要なエスケープがちょっと違うのでそこは注意してください。

5.3 実行

Overviewer.py のあるディレクトリに移動して、下記コマンドを実行してください。

```
# ./overviewer.py -c ./<作った config ファイル>
```

実行してエラーが出なければ、しばらくするとマップが生成されます！ 5.2 の **outputdir** で指定したパスの「**index.html**」を開き、マップが生成されたことを確認してください。

5.3.1 実行オプション

実行時に使えるオプションがいくつかありますので紹介します。一応他にもあるのですが、基本的には以下のオプションを覚えておけば大丈夫かと。必要なら公式ドキュメントも読んでみてください。

–no-tile-checks

デフォです。特に指定しなければこれになります、更新があったブロックだけ処理します。

–forcerender

全ブロックについて、更新されているかどうかに関わらず処理します。初期生成をもっかいやるものと思ってください。メチャメチャ時間掛かるので、設定変えた後の初回実行時などに手動で使用しましょう。

–check-tiles

各ブロックについて最新かどうかのチェックを行います。無くなってたら消します。主に**無くなったチャンクを消す時に使います**。

–genpoi

POI 生成用です。7 章で説明します。

6 ディメンションの追加

オーバーワールドだけではなく、ネザーやエンドも出力できます。また、オーバーワールドの夜間なども出力可能。出力した別ディメンションは生成した画面右上のプルダウンから切り替えて表示できます (図 3)。Daytime、Nighttime(設定によって表示名は違います) は昼と夜両方を生成している場合に表示されます。

(a) 表示メニュー　　(b) ディメンション選択

図 3: 右上プルダウンメニュー

6.1 ネザーの追加

dimension に **nether** を指定することでネザーが生成されます。4.2 章や 5.2 章で作成した config ファイルにコード 3 の通り renders を追記します。

ソースコード 3: ネザー用 renders

```
renders["nether"] = {
    "world": "survival",
    "title": "Survival Nether",
    "rendermode": nether_smooth_lighting,
    "dimension": "nether",
}
```

rendermode には必ず以下のいずれかを指定してください。他を指定した場合、天井が描写されてしまうため何も見えなくなります。

nether

照明レベルや光を無視してブロックを描写 + 天井を描写しないモードです。ネザーだと一番見やすいけどのっぺり。

nether_lighting

照明レベルとか光を反映 + 天井を描写しないモードです。それっぽくなるんだけど暗いところがベースのネザーなので何も見えなくなるかも。

nether_smooth_lighting

nether_lighting より更に滑らかな光の描写がされる……らしいです。筆者は違いがわからなかった。ちなみに nether <nether_lighting <nether_smooth_lighting の順にレンダリングが重くなります。

6.2 エンドの追加

dimension に **end** を指定することでエンドが生成されます。4.2 や 5.2 で作成した config ファイルにコード 4 の通り renders を追記します。rendermode は normal が一番わかりやすいかと。天井が無いので制限は無いですが、常にとても暗い (デフォが夜) ので lighting 系統は何も見えないかも……

ソースコード 4: エンド用 renders

```
renders["end"] = {
    "world": "survival",
    "title": "Survival End",
    "rendermode": normal,
    "dimension": "end",
}
```

6.3 夜の追加

別ディメンションだけでなく、オーバーワールドの夜を生成することもできます。**rendermode** に **night** か **smooth_night** を指定すると夜になります。4.2 や 5.2 で作成した config ファイルにコード 5 の通り renders を追記します。

ソースコード 5: 夜用 renders

```
renders["survivalnight"] = {
    "world": "survival",
    "title": "Survival Nighttime",
    "rendermode": smooth_night,
    "dimension": "overworld",
}
```

night

夜モード。全体が照明 lv0 ベースになるものと思われます。松明の明かりとかが反映されて綺麗。

smooth_night

更に綺麗な夜モード。筆者には違いがわからない……

7 genpoi を使おう

POI を利用して看板や特定の設置物の表示、マーカーの作成などを行える genpoi の説明をします。

7.1 genpoi もとい POI is 何

POI(Point of Interest) は大元の意味としては「地図上の特定のポイント」を指す単語です。minecraft においては**その座標が持っている情報**だと思っておいてください。x,y,z で示される座標に対して「ここには看板があって○○が書いてある」「ここは村の中心」とかの情報を持たせて、minecraft というシステムから参照できるようにするためのもの、といった感じです。~~資料少なすぎて正直自信がない~~

Overviewer はこの POI を探索し、指定した POI がある座標を検索し、そこの POI から取り出した情報を表示する、という機能があります。それが genpoi です。「看板を探して看板があるところには看板と内容を表示する」「ベッドがあればそこにベッドを表示する」とかできます。また、Overviewer 用に POI を設定してやることもできます。座標を指定して「この辺なんか建てる予定地」とかのマーカーを表示したりできます。

7.1.1 実際のイメージ

こんな感じ (図 4) で表示されます。見てて結構面白いんだこれが。

(a) Overviewer 側　　(b) 実際の看板

図 4: POI のサンプル

基本的に表示できるのは「マーカー」と「ちょっとした吹き出しとメモ程度の文章」になります。

7.2 設定と genpoi 実行

実際に設定してなんか表示してみましょう。作業の流れは以下の通りです。

1. 表示したい POI を定義する。フィルター関数ってので定義します。
2. ディメンションに定義した POI を表示するように指定する。
3. genpoi 実行。

7.2.1 フィルター関数の定義

まずは表示したい POI を定義します。例として看板を定義してみましょう。config の頭にコード 6 の内容を記載します。

ソースコード 6: 看板の定義

```
1 def signFilter(poi):
2     if poi['id'] == 'minecraft:sign':
3         return "\n".join([poi['Text1'], poi['Text2'], poi['Text3'], poi['Text4']])
```

これは**フィルター関数**と呼ばれており、何を表示する気なのか、どう表示するのか、を定義しています。それぞれの場所で何やってるの？を説明します。

1 行目 `def <任意の関数名>(poi):`

この定義の名前を付けています。後で表示するように指定する際、この名前を使うので関数名はわかりやすいやつにしときましょう。今回は看板のフィルター関数なので signFilter にしてます。

2 行目 `if poi['id'] == 'minecraft:<POI の ID>':`

何を探して表示する気なのかを指定しています。ブロックエンティティの名前空間 ID を指定する必要があり、看板なら"sign"です。それどうやって調べるんだよ！は 7.2.2 章で説明します。

3 行目 `return <文字列>`

吹き出しの中に表示する文字列を定義します。POI に含まれているデータとかもちゃんと指定すれば引っ張れますが、書き方に結構クセがあります。7.2.3 章で事例をいくつか紹介するので、それを元に組んでみてください。

なお、フィルター関数は複数定義を作成しても問題ありません。複数の定義を同時に表示することもできます。

7.2.2 POI の調べ方

先ほどのフィルター関数で POI に含まれているデータを普通に使いましたが、そんなもんどこに書いてあるんだという話ですね。端的に言うと Minecraft Wiki(`https://minecraft.fandom.com/ja/wiki/Minecraft_Wiki`) に書いてあります。例として**ベッド**を調べてみましょう。

1. まずは Minecraft Wiki の「Chunk フォーマット」ページ (`https://minecraft.fandom.com/ja/wiki/Chunk` フォーマット) を開きます。
2. 該当ページを開いたら「ブロックエンティティフォーマット」に移動します。
3.「ブロックエンティティ」の「ID」欄に書いてあるのが POI の ID です。
4. 更に「対応するブロック」から該当ブロックのページに移動します。
5. 対象ページの「ブロックエンティティデータ」に記載されているものが利用できるデータになります。[7)]

せっかくなので調べたベッドについても改めてフィルター関数を組んでみましょう。以下の内容でベッドを表示する準備ができますね。

ソースコード 7: ベッドの定義

```
1 def bedFilter(poi):
2     if poi['id'] == 'minecraft:bed':
3         return str(poi.get('id'))
```

7) 日本語 wiki はたまに間違っている (2022 の 12 月現在でベッドに color とかいう存在しないキーが記載されてる) ので英 wiki も見ましょう。

7.2.3 フィルター関数の事例 (作成中)

return のところには色々なパターンがあります。いくつか組んでみたので紹介します。

ソースコード 8: フィルター関数事例

```
# ブロックや中身に関わらず指定した文面を表示。
def signFilter(poi):
    if poi['id'] == 'minecraft:sign':  #ここでは看板(minecraft:sign)だが何でもok。
        return "任意の文字列を入力。"


# 看板の中身の表示。
def signFilter(poi):
    if poi['id'] == 'minecraft:sign':
        return "\n".join([poi['Text1'], poi['Text2'], poi['Text3'], poi['Text4']])


# 座標の表示。看板(minecraft:sign)を指定してしますが何にでも使えます。
def signFilter(poi):
    if poi['id'] == 'minecraft:sign':
        return ("x=%s,y=%s,z=%s" % (str(poi.get('x')),str(poi.get('y')),str(poi.get('z
          ')))


# 中身のアイテム表示。以下はチェスト(minecraft:chest)用です。
# 他のインベントリ持ちにも使えます。
def chestFilter(poi):
  if poi['id'] == 'minecraft:chest':
    # LootTableは未開封の野良チェストとかに設定されている値です。
    # これがあると中身の参照時にエラーを吐くため、表示対象から除外します。
    if "LootTable" in poi:
      return None

    # 「itemlist」という文字列の変数を立てて、そこにアイテム名(id)をひたすら繋げます。
    # Slotはチェストの位置です。左上が1番、右下に向けてどんどん番号が増えます。27まで。
    # Countはアイテムの個数。
    else:
      itemlist = ""
        for item in poi.get('Items'):
          itemlist = itemlist \
          + str(item['Slot']) \
          + ":" + item['id'].split(':')[1] \ # idの前半のminecraft:が邪魔なので消す
          + "(" + str(item['Count']) \
          + ")" \
          + "\n"
      return ("%s" % itemlist)
```

7.2.4 POI の表示指定

定義が済んだら実際に表示させます。試しに 7.2.1 章で定義した看板を表示してみましょう。**renders** に"markers: [dict(name="**<表示名>**", filterFunction=**<フィルター関数名>**)]"を書き足します (コード 9)。

ソースコード 9: POI の表示設定

```
renders["survivalday"] = {
    "world": "survival",
    "title": "Survival Daytime",
    "rendermode": normal,
    "dimension": "overworld",
    "markers": [dict(name="All signs", filterFunction=signFilter)]
}
```

name や filterFunction などの指定できるオプションについては以下の通り。

name(必須)　この名前で Overviewer の表示メニューとして出てきます。図 5 の赤枠の部分ですね。なんとデフォルトで表示 ON/OFF ができる！

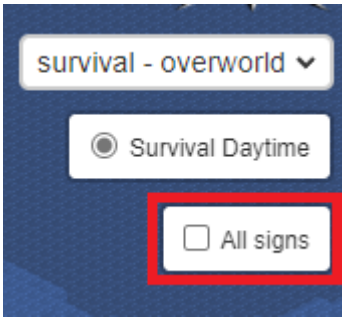

図 5: genpoi 表示メニュー

filterFunction(必須)　7.2.1 章とかで組んだファイルター関数の関数名を入れます。

icon　Overviewer に表示される際のアイコンを指定できます。デフォルトは看板。icon=**ほにゃらら**.png とかで指定します。なお、**アイコン画像は出力するマップの index.html と同じディレクトリにある**必要があります。自作用のテンプレがあるので 7.4 章で紹介します。

createInfoWindow　クリックした際に吹き出し表示するかどうかを指定できます。デフォルトは True で、False を明示すると表示されなくなります。

showIconInLegend　デフォルトは False で、True を指定するとアイコンの凡例が表示されるようになります。図 6 のような感じに。

checked　デフォルトは False で、表示 ON/OFF のデフォルトを指定できます。True にすると最初から ON になります。

図 6: showIconInLegend 有効時

なお複数のフィルター関数を同時に有効化することもできます。カンマで区切って dict(〜) を複数書いてあげれば ok です (コード 10)。看板だけではなく、7.2.2 章で定義したベッドも表示しようとしています。

ソースコード 10: markers 複数

```
"markers": [dict(name="All␣signs", filterFunction=signFilter),
            dict(name="All␣beds", filterFunction=bedFilter)]
```

7.2.5 genpoi 実行

ここまで準備したら、あとはレンダリングだけです。以下のコマンドを実行しましょう。

Windows の場合

```
> overviewer.exe -c "＜ config ファイルのパス＞" --genpoi
```

linux の場合

```
# overviewer.py -c "＜ config ファイルのパス＞" --genpoi
```

重要なのは「--genpoi」をオプションで付けることです。このオプションを付けると**通常の world の出力を行わず、POI だけを生成**します。もし world 生成もセットで行うのであれば、下記のように 2 回に分けて実行する必要があるので注意してください。

```
> overviewer.exe -c "＜ config ファイルのパス＞"
> overviewer.exe -c "＜ config ファイルのパス＞" --genpoi
```

7.3 自作 POI 設定と表示

minecraft 内の要素を表示するだけでなく、自前でマーカーを設定して表示させることも可能です。フィルター関数にコード 11 を、renders にコード 12 を挿入します。図 7 が出力例となります。[8)]

ソースコード 11: 自作 POI(フィルター関数)

```
def testFilter(poi):
    if poi['id'] == 'test':
        return str(poi.get('name'))
```

ソースコード 12: 自作 POI(renders)

```
'manualpois':[
    {'id':'test',
    'x':200,
    'y':64,
    'z':200,
    'name':'testPOI'}],
'markers': [dict(name="test", filterFunction=testFilter, icon="markertest.png")]
```

図 7: 自作 POI

manualpois で指定した name の値をフィルター関数の return で指定して吹き出しに出力しています。なので、ここで manualpois の name の内容を書き換えることで吹き出しの中身も変えることができますね。「testPOI」を「this is test」とかに変えてやれば吹き出しにも「this is test」と出ることでしょう。

8) アイコンは自作したやつです。

7.4 アイコン作成

POI に使うアイコンは、デフォの看板以外は自作したり著作権フリーを引っ張ってくる必要があります。幸い、自作についてはテンプレを用意してくれているので、それを改造するだけで済むようになっています。テンプレートの場所は Overviewer をインストールしたディレクトリ内の、`overviewer_core\data\web_assets\markers` 内にある marker_base_plain.svg と marker_base_plain_red.svg です。svg の扱いに慣れた人はチャチャッと編集してバンバン使っていきましょう。svg なんぞという人は以降の内容を読んでみてください。

7.4.1 Inkscape インストール

まずは svg 編集ができるソフトを調達します。今回は Inkscape を導入しましょう。以下の手順でインストールしてください。※ Windows 用の手順になります。

1. `https://inkscape.org/ja/` にアクセスします。
2.「DOWNLOAD」にカーソルを合わせて「Current Version」を選択します。
3.「Windows Installer Package」を選択します。
4. ダウンロードされた msi ファイルを実行します。
5.「Next」「Next」「Install」でインストール開始されます。必要な場合は適宜「Custom Setup」画面でインストール先を変更するなどしてください。
6.「Finish」でインストールを終了します。

7.4.2 テンプレートの編集

絵心がないので png 生成の手順だけ記載します。編集は各自調べて頑張ってください……

1. インストールした Inkscape を起動します。
2.「Welcome!」表示された窓が開きます。「絵を描く」に移動します。
3.「他のファイルを参照」を選択した状態で「開く」を選択します。
4. marker_base_plain.svg か marker_base_plain_red.svg を選択してください。
5. 開かれた画像を適宜編集します。終わったら Ctrl+Shift+E を押下します。
6.「エクスポート」右上のプルダウンで png を指定します。
7.「エクスポート」左上のフォルダのマークを選択し、保存先と保存したいファイル名を指定します。
8.「エクスポート」を選択して下さい。png 生成されます。

8 カスタムレンダーモードとオーバーレイを使おう

config の rendermode について、突っ込んだ設定をするための説明を行います。出力されるマップに幅が広がります！

8.1 カスタムレンダーモード is 何

ここまで rendermode に指定してきた「normal」とか「nether」というのは、実は裏で用意されているオプション[9]の組み合わせでできており、そのオプションを直接指定して自分なりの rendermode を指定することが可能です。これがカスタムレンダーモードです。

8.2 カスタムレンダーモードの書き方

シンプルな形としては、コード 13 のような形で rendermode に使用したいオプションを指定します。これは通常の出力「Base()」に液体を描写しない設定「NoFluids()」を追加したものです。また、コード 14 のように、事前に定義しておいて定義を rendermode に指定することも可能です。定義を使い回す場合に有効です。図 8 が出力例です。水が全く表示されていませんね。

ソースコード 13: カスタムレンダー指定

```
"rendermode": [Base(), NoFluids()],
```

ソースコード 14: カスタムレンダー指定その 2

```
myrendermode = [Base(), NoFluids()]

～～

# world・title・dimensionはこれまで通り指定します。
# rendermodeに定義した「myrendermode」を指定してください。
renders["survivalday"] = {
    "world": "survival",
    "title": "Survival␣Daytime",
    "rendermode": myrendermode,
    "dimension": "overworld"
}
```

9) レンダーモードプリミティブと呼ぶそうです。

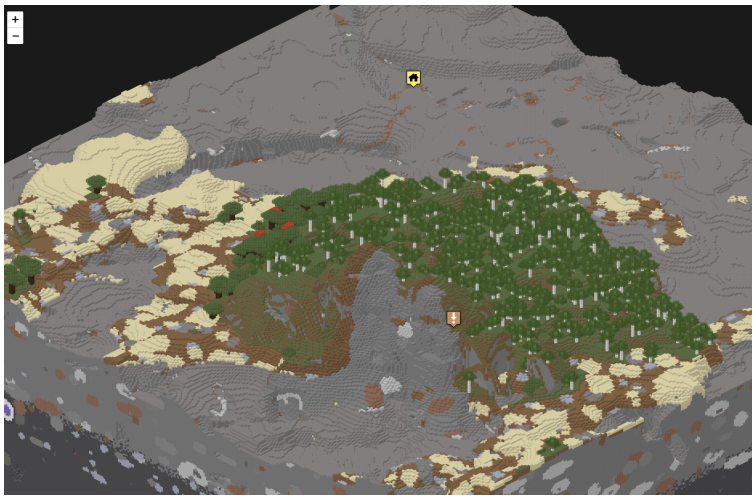

図 8: カスタムレンダー出力例

8.3 オーバーレイ is 何

特定の要素を座標で色付けしただけのマップを生成して、それを他のマップに被せて表示するような機能です。スライムが出てくる座標に色付け、とかできます。コード 15 のように書きます。

ソースコード 15: オーバーレイ指定

```
1
2  # 普通に生成するrenders。ここで入力した"survivalday"という名前を後で指定します。
3  renders["survivalday"] = {
4  〜省略〜
5  }
6
7  # world・titleはいつも通り指定してください。
8  renders["overlay"] = {
9      "world": "survival",
10     "title": "Overlay",
11     "rendermode": [ClearBase(), SlimeOverlay()],
12     "overlay": "survivalday"
13 }
```

rendermode 「ClearBase()」と使いたいオーバーレイ用オプション、という指定になります。「ClearBase()」は背景を透明にするという処理で、生成したマップを色付け部分だけにしてくれる感じですね。

overlay 被せる先のマップを指定します。被せたいマップの **renders** の名前を入れてください。

図 9 が出力例です。海の上の四角形や、木の上になんかかかっているのがスライムチャンクを示す色付けです。

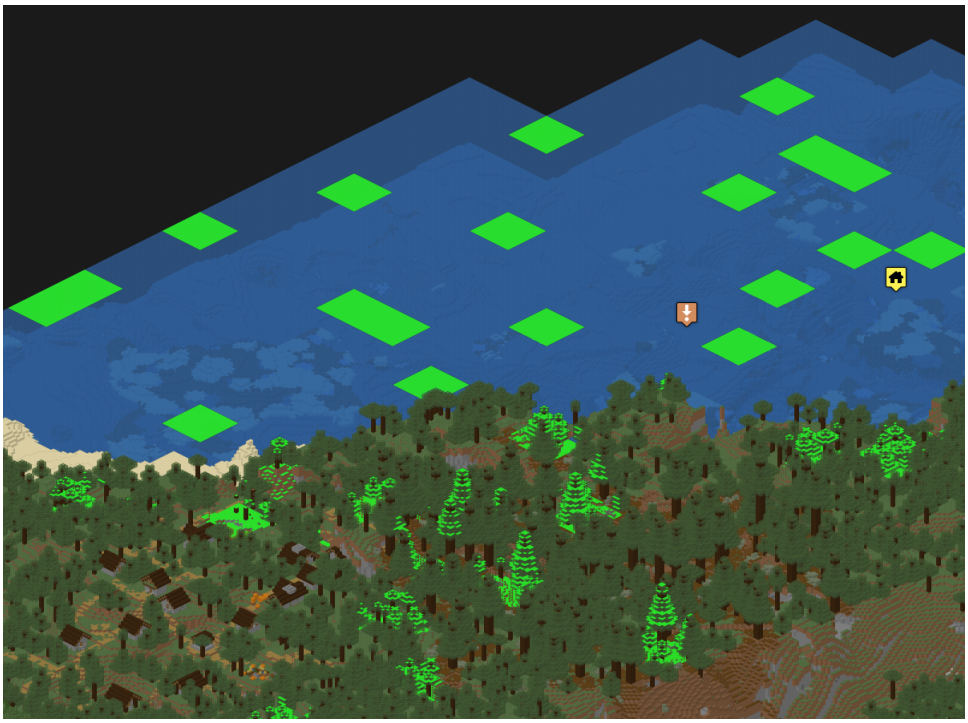

図 9: オーバーレイ出力例

8.4 オプション (レンダーモードプリミティブ) 一覧

どんなオプションがあるか紹介していきます。

8.4.1 通常出力用

Base()

通常の出力です。**オーバーレイ以外は必須**です。`Base(biome=False)` とするとバイオーム毎のカラーリングが無効化されます。

nether()

天井を消去します。ネザーをレンダリングする際、指定しないと天井だけが表示されるマップが生成されます。

HeightFading()

高さが一定ラインを越えるまでブロックに色を付ける……らしいです。~~ぶっちゃけよくわからなかったですすみません~~

Depth()

「Depth(min=10,max=30)」のようにすることで、指定の高さのブロックだけを表示するようにします。指定可能な範囲は 0〜255。マイクラ v1.18 以降の高さに非対応…… ~~あと動作的にこれは Depth ではなくない？~~

Exposed()

空気に触れているブロックだけ表示されるようになります。反転させることも可能で「Exposed(mode=1)」にすると空気に触れていないブロックだけ表示されます。

NoFluids()

液体を非表示にします。図 8 でやりましたね。

EdgeLines()

ブロックの裏側の境界に線が引かれます。「EdgeLines(opacity=1.0)」のように値を指定すると、線の太さが変わります。0 から 1.0 まで。

Cave()

日光が当たっているブロックを非表示にして、地下洞窟を表示します。「Cave(only_lit=True)」だと照明で照らされている地下だけ表示されます。~~ヒカリゴケ爆増でメチャメチャ使いにくくなった~~

Hide()

特定のブロックを非表示にします。`Hide(blocks=[1,2,3])` のような形で指定します。非表示にしたいブロックは数字の ID で指定する必要があります。ブロック ID は `https://minecraft.fandom.com/ja/wiki/データ値/JavaEdition/平坦化前/ブロック ID` を参照してください。

DepthTinting()

高さに合わせて色付けされます。底に近ければ暗く赤くなり、空に近ければ明るく緑になっていきます。

Lighting()

照明レベルが ON になります。`Lighting(strength=0.5)` で影の強さを調整 (0〜1.0・デフォ 1.0)、`Lighting(night=True)` で日光オフ (夜モード)、`Lighting(color=True)` で影に色が付きます。それぞれをカンマで区切って同時に指定することも可能です。

SmoothLighting()

Lighting と基本は同じです。こちらの方が滑らか。~~見てもイマイチわからない~~

8.4.2 オーバーレイ用

ClearBase()　オーバーレイの場合は Base() ではなくこちらを使用してください。

SpawnOverlay()　照明がないなどで MOB がスポーンする可能性のあるブロックに色付けされます。`SpawnOverlay(overlay_color=<赤>,<緑>,<青>,<透明度>)` で色を変更することも可能です。

SlimeOverlay()　スライムチャンクに色が付きます。SpawnOverlay 同様、overlay_color が使えます。

MineralOverlay() 鉱石が埋まっている座標 (x,z) の地表部分に色を付けます。`MineralOverlay(minerals=[(<ブロック ID>,(<赤>,<緑>,<青>))])` で鉱石毎に色を設定することが可能です。

StructureOverlay() 「ブロック A を基準にして、ここにこのブロックがある場合、この色で色付けする」という動作をするとのことです。公式 doc によると丸石の上のレール、とか。~~ぶっちゃけよくわからなかったですマジですみません~~

BiomeOverlay() バイオーム毎に色が付きます。`BiomeOverlay(biomes=[(`**バイオーム名**`,(<赤>,<緑>,<青>))])` で色を指定可能。**2022/12 現在、minecraft 1.18 以降は正常動作しません。**

HeatmapOverlay() 「レンダリングした日付から見て X 日以内に訪問されたかどうか」を基準にチャンクに色付けします。訪問された日付が近ければ近いほど明るい色になります。**t_invisible**(色が消える日付) と **t_full**(色が最も明るくなる日付) を指定できます。デフォルトは 30 日。指定が結構面倒なのでコード 16 に例を載せます。例は 2 日訪問しなかったら色が消えますね。

ソースコード 16: HeatmapOverlay の例

```
HeatmapOverlay(
    t_invisible=int((t_now - timedelta(days=2)).timestamp()),
    t_full=int(t_now.timestamp()))
```

8.5 組み込みレンダーモード

導入の時に指定した「normal」や「night」なども実は 8.4 の組み合わせで作られています。表 1 の通りです。

normal	Base(), EdgeLines()
lighting	Base(), EdgeLines(), Lighting()
smooth_lighting	Base(), EdgeLines(), SmoothLighting()
night	Base(), EdgeLines(), Lighting(night=True)
smooth_night	Base(), EdgeLines(), SmoothLighting(night=True)
nether	Base(), EdgeLines(), Nether()
nether_lighting	Base(), EdgeLines(), Nether(), Lighting()
nether_smooth_lighting	Base(), EdgeLines(), Nether(), SmoothLighting()
cave	Base(), EdgeLines(), Cave(), DepthTinting()

表 1: 組み込みレンダーモード一覧

9 24h サーバーにおける運用

シングルで自分の好きな時に Overviewer を実行しマップ生成する分には、空いた時間に手で都度実行すれば問題ありません。しかしマルチサーバーで Overviewer を運用する場合、以下のような問題が出てきます。

- レンダリングがかなり重いため、minecraft 本体 (以降本体) と同時実行した場合に本体に影響が出る可能性がある。特に調整段階で何回もレンダリングする場合、本体と同じマシンでやるのは憚られる。
- マルチ前提とした場合、自分の好きな時に実行するのではなく自動で定期実行する必要がある。[10)]
- マップが大きくなってくるとレンダリングに非常に時間がかかる。[11)]定期実行が次までに終わらないのが怖くなる。
- マルチ用に生成する場合公開する手段が必要になる。具体的には http サーバを立ててネット経由で見えるようにしたくなる。

じゃあどうするのかというところで、Overviewer は minecraft の MOD ではなく外部ツールであるということを利用し、筆者環境では本体と Overviewer でマシンを分けています。Overviewer が本体に与える影響を最小限にすることが狙いです。この環境の説明と注意事項等を以降で説明していきます。

10） まあ身内鯖とかなら手動でも別にいいかもしれない。

11） 筆者の身内鯖でテストして、10 万タイルで 2 時間弱くらいかかった。タイル数は初期生成マップで 2000 くらいなので、ちゃんとプレイヤーを統率できてるならもうちょい何とかなるだろうけど……

9.1 必要なもの

マシンを分けるにあたり、本体と Overviewer 以外に必要な要素が以下となります。

マシン 2 台
当然ではありますが。物理を 2 台用意するのは面倒なので、何らかの仮想環境を使うのがいいかなと思います。

OS2 つ
安く済ませるなら linux2 つ。win のライセンスが湯水のようにあるなら少なくとも Overviewer は win にしちゃうのがいいんじゃないかな！お金かけないなら頑張って Linux。

作業用端末
普段使いしてる windows とかで ok。Overviewer の運用を制限して兼任させるという選択肢もあります。

2 台間のファイル転送
Overviewer 側のマシンが本体の world データを参照する必要があります。このデータ連携に何らかの処理が必要。

回線　恐らく自宅サーバ勢にとって最大のガチャ。ゴールデンタイムに重くなるとかなら自宅サーバはやめた方がいいと思います。VPS とか使う最大の利点はここを買えるとこじゃないかな……

マシン 2 台の固定 IP もしくはドメイン名
VPS2 台とかにすると面倒くさそう。

9.2 マシンの準備

マシンの用意の仕方もまあ色々あります。

物理マシン
平成を信じろ。組んでしまえばまあ割と後は楽です。色々と自分の都合に合わせやすいのは利点。光熱費はかかるし物理スペースも要ります。

- スペックや運用、環境などを自分の都合に合わせやすい。容量拡張が VPS とかより柔軟なのと、電源 ON/OFF を物理的に握れるのは大きいです。何かあった時に取れる選択肢も多い。総じて~~自分のやる気があれば~~柔軟性が高いです。
- 知識面のハードルが低め。[12)]
- とにかく金がかかる。初期費用だけじゃなくて光熱費で恒常的に金がかかってきます。課金状況が分散して全体の費用が見えにくいのも欠点。

12)　自作の場合はちょっと話が変わってくる。BTO とかで用意するのがいいかなあ……

- 回線ガチャ次第で使い物にならない場合があります。自宅サーバ最大の敵。
- 何かあった時 #誰かのせいにしたいが自分の顔しか思い浮かばないの画像をここに貼る

物理ハイパーバイザ 1 台+仮想マシン

VMware Hypervisor(ESXi) とか KVM とかで構成して、その上に仮想マシンを 2 台立てる方法です。筆者はこれ (VMware Hypervisor)。

- 物理マシン複数台よりも更に柔軟性があります。大体の場合今の物理マシン 1 台はオーバースペックなので、それを好きなように分割して使える状態になります。恐らく最もやりたい放題できる構成だと思います。
- 物理複数台よりは確実に安いです。1 台しか用意しないし。光熱費とかも当然下がる。HDD の冗長化とかしてもまだ安い。
- ハードル高め。仮想化というものに対するイメージ力が必要になってきます。どこがどう物理ハードと関わるのかとか考えられないとならないし大半の設定が物理的に見えない。
- 物理マシンが死んだら全部死にます。その為に色々冗長化すると今度は金と技術が……その内企業レベルのオンプレ構築に……
- 回線ガチャには勝てない。各社もうちょっとどうにかしてくださいよォ！
- 何かあった時 #誰かのせいにしたいが自分の顔しか思い浮かばないの画像をここに貼る

VPS

恐らく最も手軽で安価であろう環境。サーバやったことないならまずはこっから開始するのがいいと思います！実は殆どやったことなかった。

- クッソ安い。4GB2 コアで 1 台 2000〜3000 円/月くらい。本体用は 8GB 欲しいけどまだ現実的なライン。ちなみに筆者の物理サーバは 24h 稼働で電気代が 3000 円/月以上かかってるらしいです。周辺機器も考えると……
- ハード面の障害があった場合に VPS サービス提供側のせいにできます。誰かのせいにできる！
- 回線ガチャから解放されます。ここは回線難民のフロンティア。
- 拡張性に乏しい。ちゃんと考えておかないと特に HDD 容量不足で死ぬでしょう。
- ハード面の障害があった場合に自分ではどうにもできません。誰かのせいにした結果がこれだよ。

クラウドサービス

死ぬほど高いので試せません！ノーコメントで！

9.3 OS の選定

VPS だと勝手に決まるので何も気にしなくてよいです。自宅サーバの場合は自前で用意する必要あり。

Windows

とにかく使いやすい。動作の最適化とかは Linux に劣りますが、楽なのは正義。使うならパッケージ版買うか VPS で。

- Overviewer が動かしやすいです。最新環境への最適化が間に合ってないツールなので、互換性の神たる Windows に導いてもらうのが本当に楽。
- 本体とか Overviewer 動かす専用サーバとした場合に、余計な物が動きすぎていると感じる場合はあります。突き詰めるなら Linux に移動。
- 有料。home でも 2 万弱する。バンドルとかだと安いけどサーバ運用する場合バンドル特有の制限が足を引っ張る場合あり。
- 本体にバッチからコマンドを送ることができなさそう。world データコピーの自動化ができないので、**Overviewer 運用する場合 minecraft 本体を windows で動かすのは NG** です。

Linux

Overviewer 動かす場合キレそうになるくらい使いにくいです。最適化するなら仕方ない。各ディストリビューションの公式からダウンロードして使いましょう。

- 要らない機能をかなり削ぎ落とせるので動作の最適化ができます。できるなら本体は Linux で動かす方がいい。
- 個人レベルなら無料の範囲で運用可能。VPS とかでも Linux マシンの方がメチャメチャ安いんですよね。
- Windows しか触ったことがない場合間違いなく使いにくいです。CLI 操作ができるのは前提。
- 互換性がカス。開発が最新環境に追い付いていない Overviewer は導入から地獄を見ます。
- CentOS が死んだので最新環境を使うなら rhel を無償利用の範囲で使うことになります。もしくは Fedora とか Ubuntu 系とか。筆者は rhel 慣れしてるので CentOS とか rhel しか殆ど使ったことないんですけどね！
- 本体を linux で稼働させる場合、必ず screen を利用して起動してください。minecraft の管理コマンドを linux の自動処理に組み込むのに必要。

MAC

~~#りんごのアンチじゃったかの画像をここに貼る~~ サーバとして使ったことないのでノーコメントで……

9.4 実際のマシンの用意

計画が立ったら minecraft server 本体と Overviewer 用にサーバを立てましょう。minecraft サーバの立て方は本筋じゃないので、Minecraft Wiki の「チュートリアル/サーバーのセットアップ」を参照してください。Overviewer のインストール手順は 4 章 (windows 用) と 11.4 章 (rhel 用) を読み返してください。

9.5 サーバ 2 台間のファイル転送設計

マシンと OS を決めてマイクラ本体の導入くらいまでは済んだでしょう！次に考えるのはファイル転送設計です。Overviewer は本体の world データを元にマップを生成するので、Overviewer が動くサーバから本体の world データを何とかして見なければいけません。つまりファイルを転送するなり共有するなりする必要があります。その際に考慮する点は以下の通り。

ファイル参照の方向
: 本体が Overviewer に転送する・Overviewer が本体に取りに行く、などのパターンがあります。これによって使う手法やコピーした world データの置き場所が変わります。

参照用データの生成
: 稼働中の world データはリアルタイムで更新されているので、これを直接 Overviewer から見るのは非推奨です。このため、world データを一回どっかにコピーしたものを Overviewer から見る必要があります。

ファイル参照の方法
: FTP や SCP で転送・NFS や SAMBA で共有・NAS で共有など。

この辺のイメージを付けた上で、どうするかを事前に決めておきます。決めるための話を以降のページでしていきます。~~行き当たりばったりでやると後で泣きます。泣いた。~~

9.5.1 参照用データの生成

次は Overviewer が参照するための world データ生成を行います。以下の順で実行。なお、OS 選定でも述べましたが windows で本体を動かしている場合自動化は不可能と思われます。

1. 以下の通り、マイクラ側で管理コマンド save-all を実行。これで実行時点の状況が HDD の world データに反映されます。
   windows　minecraft server のコンソールから *save-all* を実行してください。

   linux　screen を利用している前提の実行方法です。以下を実行します。
   ```
   # screen -p 0 -S <起動時に指定したscreenの名前> -X eval 'stuff "save-all"\015'
   ```

2. 以下の通り、マイクラ側で管理コマンド save-off を実行。自動保存を一時的に止めて、HDD の world データが更新されないようにします。
   windows　minecraft server のコンソールから *save-off* を実行してください。

   linux　screen を利用している前提の実行方法です。以下を実行します。
   ```
   # screen -p 0 -S <起動時に指定したscreenの名前> -X eval 'stuff "save-off"\015'
   ```

3. world データを自分で決めた任意のディレクトリにコピーする。このデータを転送したり Overviewer から見に来たりしてマップ生成することになります。
4. 以下の通り、マイクラ側で管理コマンド save-on を実行。自動保存を再開します。
   windows　minecraft server のコンソールから *save-on* を実行してください。

   linux　screen を利用している前提の実行方法です。以下を実行します。
   ```
   # screen -p 0 -S <起動時に指定したscreenの名前> -X eval 'stuff "save-on"\015'
   ```

linux であれば起動停止と合わせてスクリプトにしてしまう方がよいと思います。10 章でスクリプトの例を提示しています。

9.5.2　world データの転送・参照

world データを無事コピーできたでしょうか。できたのであれば、それを Overviewer に見せてあげるのが次のステップです。いくつか手段がありますので、筆者のできる範囲で紹介していきます。※それぞれの導入は本筋より長くなりそうなので割愛します……

FTP や SCP による転送 (図 10)　転送先のマシンに直接ファイル叩き込んだり、転送元のマシンに直接ファイル取りに行ったりする方法です。linux であれば vsftpd、windows であれば IIS の FTP サーバを導入します。多数のファイルをコピーする場合に tar などでまとめないといけないのが少々ネック。

ファイル共有 (図 11)　参照用 world データを転送元・転送先両方で見えるようにする方法です。Linux 間であれば NFS、Linux-win 間なら samba、win 間であればファイル共有でそれぞれ調べてみるとよいでしょう。イメージを掴むのは FTP よりちょっと面倒ですが、多数のファイルを連携するのには向いています。

nas による共有 (図 12)　金があるなら nas で共有するという手もあります。nas というものがそもそもそういう目的にも適している代物なので多分一番楽です。問題は金。

外付け HDD とか USB メモリ (図 13)　今令和ぞ。

~~TB 単位とかでしかもインターネット経由とかだとこっちの方が速い。~~

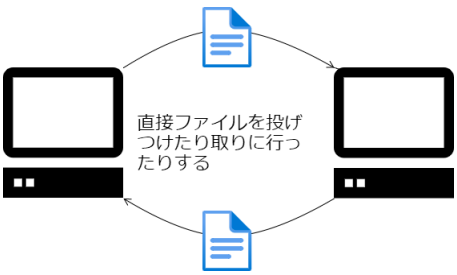

図 10: FTP 転送・SCP 転送

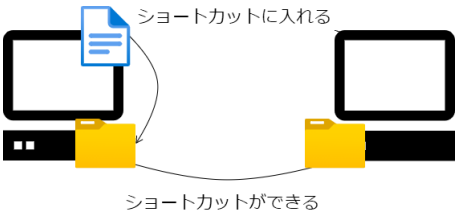

図 11: ファイル共有

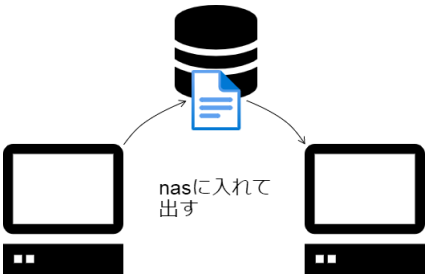

図 12: nas による共有

図 13: スニーカーネットとかいうやつ

9.6 Overviewer 側の config 調整

ファイル転送が上手くいけば、Overviewer 用のマシンで wolrd データのコピーが見えていると思います。後は Overviewer の config で *worlds* に見えた world データのあるディレクトリを指定し、Overviewer のレンダリングを実行してください。

9.7 http サーバの用意

先述していますが、マルチ参加者みんなでマップを見たい場合は公開用に http サーバを立てる必要があります。Overviewer が動作しているサーバで同時に動かす形で問題ないと思います。

9.7.1 http インストール

windows

以下手順でインストールします。

1. Win + R で「ファイル名を指定して実行」を起動します。
2. **OptionalFeatures** を実行します。
3.「インターネット インフォメーション サービス」に新規にチェックを入れます。「OK」選択したら再起動。
4. Win + R で「ファイル名を指定して実行」を起動します。
5. **inetmgr** を実行します。「インターネット インフォメーション サービス (IIS) マネージャー」が起動すれば成功です。

Linux(rhel 系)

基本的には dnf コマンドで一発です。実際に実行する際は root でやるか sudo 入れてください。インストール後は起動と自動起動の有効化も行っておきます。

```
# dnf install httpd
# systemctl start httpd
# systemctl enable httpd
```

9.7.2 http 設定

設定開始前に、Linux/Windows どちらであってもどのディレクトリを Web ページとして公開するのかを事前に決めておいてください。Overviewer 専用にするのであれば、Overviewer の **worlds** に指定したディレクトリを指定するのがよいでしょう。

Windows

以下の手順で設定します。

1. Win + R で「ファイル名を指定して実行」を起動します。
2. **inetmgr** を実行します。
3.「サイト」→「Default Web Site」→「基本設定」の順で選択します。
4.「物理パス」を自分の使いたいディレクトリにして「OK」を選択します。
5. `http://localhost/` でエラーが出ないことを確認します。

Linux(rhel 系)

以下の手順で設定します。

1. httpd の設定ファイルを開きます。

```
# vi /etc/httpd/conf/httpd.conf
```

2. コード 17 の通り修正します。

ソースコード 17: httpd.conf 設定

```
1 # 下記の記載をWebページとして公開したいディレクトリに変更します。
2 DocumentRoot "/var/www/html"
3 ～略～
4 # 下記の記載をコメントアウトします。※文字化け防止
5 #AddDefaultCharset UTF-8
```

3. DocumentRoot で指定したディレクトリの権限を調整します。[13)]

```
# usermod -G apache <Overviewer の実行ユーザ名>
# chown apache:apache <Documentroot で指定したディレクトリ>
# chmod 775 <Documentroot で指定したディレクトリ>
```

4. 他のマシンで `http://<http サーバの IP>` でエラーが出ないことを確認します。

後は設定したディレクトリに Overviewer のマップを生成しましょう。これで自分のおうちからは見えるようになったはずです。

9.8 外部公開について

この設定やってるような人なら minecraft を公開してますよね！というわけで DNS 周りは全部省略します。しかし、**おうちサーバ環境だとすると本体用と http 用で二台マシンを立てた場合のポート転送設定が必要になります。**10.3 章で設定例を提示していますので参考にしてみてください。詳細な設定はルータやネットワークによるので各自確認としてここでは割愛します。

13) 流石に 777 は後で面倒くさいので避けた。自己責任の元なら 777 にしてやればユーザのグループ設定とか不要。

9.9 運用の自動化

ファイル転送や Overviewer の実行を自動化し、勝手に更新されるようにしましょう。

9.9.1 Windows

必要な処理を bat にまとめてタスクスケジューラで動かすことになります。このため、ファイル転送はコマンドで実行できるようにしておく必要があることに注意してください。

1. バッチファイルを作成します。コード 18 を「<任意の名前>.bat」で保存します。

ソースコード 18: バッチのイメージ

```
@echo off
<ファイル転送処理　ftpのスクリプト叩いたりrobocopyでコピーしたり>
cd /D <Overviewerのフォルダ>
<Overviewerの実行コマンド>
<genpoiの実行コマンド>
```

2. タスクスケジューラに登録します。ざっくり手順を示します。コマンドプロンプトで以下のコマンドを実行します。実行タイミングは表 2 のような指定が可能です。

   > schtasks /create /tn <タスク名> /tr <バッチファイルのパス> /sc <実行タイミング>

minute /mo X	X 分毎に実行します。
hourly /mo X	X 時間毎に実行します。分はタスク作成時刻になります。
hourly /mo X /st mm:ss	mm:ss に初回実行し、以降 X 時間毎に実行します。
daily /st mm:ss	毎日 mm:ss に実行します。

表 2: schtasks：実行タイミング設定

windows の実行スケジュールの管理は GUI(タスクスケジューラ) でもできます。運用は各自でたしかみてくれ！

9.9.2 Linux(rhel 系)

ファイル転送と Overviewer の実行をシェルスクリプトにまとめ、cron で定期実行するのが基本となります。NFS 構成のものを 14 章で例示しますので、そちらをご参照ください。

10 構成事例：筆者の構成

じゃあ実際にどんな感じになるのかということで、筆者の環境を提示します。

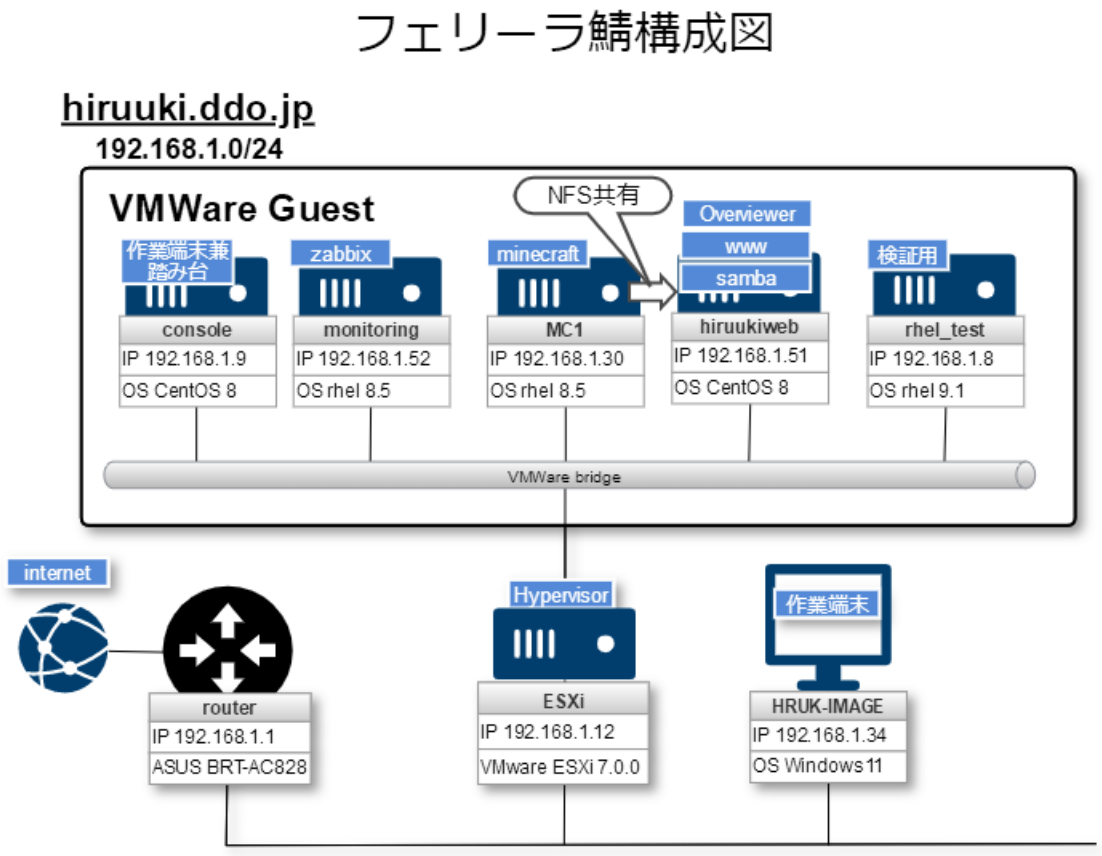

図 14: 筆者環境の構成図

ESXi に仮想マシンを載せてそこで殆どを動かしています。作業する時は脇にある windows から ESXi 内の作業用マシンに入り、そこから各マシンに接続する、という形を取っています。

各マシンのスペックが以下の通り。

esxi	
メモリ	32GB
CPU	i7-4770S(4C/8T)
ストレージ	2TB*2 本

表 3: スペック：esxi

ホスト名	MC1	hiruukiweb	monitoring	console	rhel_test
メモリ	8GB	4GB	8GB	4GB	4GB
CPU	2 コア	4 コア	2 コア	1 コア	2 コア
ストレージ	200GB	200GB	200GB	100GB	200GB

表 4: スペック：仮想マシン

各マシンの役割が以下の通りです。

esxi　仮想マシンを載せるベースです。実は windows のお下がりパーツ寄せ集めでできてる。警告出るし非推奨だけど便利なので SSH を ON にしている……

MC1　minecraft 本体を動かしています。メモリ多めの割り当てなのは RAMDISK を使うからです。hiruukiweb(Overviewer) との連携については MC1 側で NFS を設定し、hiruukiweb 側でマウントして対応しています。

hiruukiweb　Overviewer と httpd とファイルサーバをやっています。http 接続できるエリアを samba 設定入れて windows から見えるようにしておくのは非常に便利。クソみたいなホスト名なのはこいつだけ前々から使い回してるからですね……

monitoring　zabbix で各マシンを監視しています。今回は説明しませんが **minecraft のログを監視して Discord に死亡者のログを流す謎の機能**を作り込んであります。好評。

console　踏み台です。本格的に作業する時はこのマシンに接続して、ここから更に他のマシンにアクセスするようにしています。なのでこいつは GUI 環境も入れてる。作業ログをここで取れるようにしておくといいかもしれません。~~筆者ができてなくて後悔している~~

rhel_test　検証環境です。本格的に公開する気はないけど適当にテストしたいなって時に使い潰します。何も考えずに使うのでゴチャゴチャだしよく OS がリセットされる。

10.1 MC1 の設定など

普通にマイクラ入れてる以外にやってることを箇条書きで記載しておきます。

- 本体は RAMDISK 上で稼働させています。df -h 叩いたら tmpfs が 3.8GB だったので多分 4GB 割り振ってる。RAMDISK のデータは起動時・停止時・本体の定期バックアップ実行時に RAMDISK じゃないエリアにコピーさせています。この処理が Overviewer 用の world データ生成も兼任しています。
- minecraft が 1.17 以降かつ CentOS8 の場合、適当に Java を入れても動かないという罠があります。openJDK の 16 を明示的にインストールしてやる必要がある。
- 本体を動かす補助として screen コマンドを入れてます。外から save-all を叩くのに必須。
- NFS サーバ入れてます。exports ファイルは以下のような内容になっています。MCtaihi というのは RAMDISK に置いてるファイルを一時的に退避しているところですね。

ソースコード 19: MC1 の/etc/exports

```
1 /home/MC/MCtaihi 192.168.1.51/24(rw,no_root_squash)
```

- バックアップ用に rsync を入れてます。
- monitoring(zabbix) から監視するため、zabbix agent2 を入れる・10050 ポートを開ける・本体のあるディレクトリを 750 にして zabbix ユーザを minecraft ユーザのグループに所属させる　を行っています。

10.2 hiruukiweb の設定など

ここが今回の本題ですね。Overviewer の前にまずは本体の設定。

- CentOS8 なのは以前のシステムを使い回しているから。もう死んでるので本当は rhel に移行したいんですけどね……
- 複数の機能を兼任しているのでコア数は多め。メモリはそんなに使わないので 4 に抑えてます。i/o より計算力かなと。
- httpd を動かしています。Overviewer を公開する以外に、サーバ利用者向けの案内も適当に html 書いて公開しています。
- samba を動かしています。先述の通り、windows 端末から http 公開部分が見えるとものすごく便利なんですよ。http 公開部分に他のマシンに送りたいファイルを放り込んで wget で引っ張るといった雑な運用が可能になります。
- Overviewer が動いています。設定は後述。

10.2.1 Overviewer 設定

- Overviewer の config はコード 20 です。かなり見た目重視でレンダリングがクソ重く、forcerender は禁止オプションと化しています。
- Overviewer を実行する際はコード 21 のシェルスクリプトを実行します。world データを取りに行く動作とセットにしていますね。MC1 のデータを参照する期間を極力小さくしています。
  ~~詳細は覚えてないんだけど昔マウントしっぱなしにしてトラブった覚えが~~
- コード 21 のシェルスクリプトを cron に設定して自動実行させています。cron の設定はコード 22 を参照ください。

ソースコード 20: config ファイル (筆者環境)

```
def signFilter(poi):
    if poi['id'] == 'Sign' or poi['id'] == 'minecraft:sign':
        return "\n".join([poi['Text1'], poi['Text2'], poi['Text3'], poi['Text4']])

worlds["survival"] = "/var/www/html/overviewer/mc_world/world"

renders["survivalday"] = {
    "world": "survival",
    "title": "Survival Daytime",
    "rendermode": smooth_lighting,
    "dimension": "overworld",
    "markers": [dict(name="All signs", filterFunction=signFilter)]
}

renders["survivalnight"] = {
    "world": "survival",
    "title": "Survival Nighttime",
    "rendermode": smooth_night,
    "dimension": "overworld",
    "markers": [dict(name="All signs", filterFunction=signFilter)]
}

renders["survivalnether"] = {
    "world": "survival",
    "title": "Survival Nether",
    "rendermode": nether,
    "dimension": "nether",
    "forcerender": True,
    "markers": [dict(name="All signs", filterFunction=signFilter)]
}

renders["end"] = {
    "world": "survival",
    "title": "Survival End",
    "rendermode": normal,
    "dimension": "end",
    "forcerender": True,
    "markers": [dict(name="All signs", filterFunction=signFilter)]
}

renders["spawnoverlay"] = {
    "world": "survival",
    "title": "spawn overlay",
    "rendermode": [ClearBase(), SpawnOverlay()],
    "overlay": ["survivalday"]
}

renders["slimeoverlay"] = {
    "world": "survival",
    "title": "slime overlay",
    "rendermode": [ClearBase(), SlimeOverlay()],
    "overlay": ["survivalday"]
}

outputdir = "/var/www/html/overviewer/ov_map"
texturepath = "/var/www/html/1.17.jar"
```

ソースコード 21: ov 用シェル (筆者環境)

```
#!/bin/bash

mount -t nfs 192.168.1.30:/home/MC/MCtaihi /mnt/MC1/
rsync -av -delete /mnt/MC1/ /var/www/html/overviewer/mc_world/
umount /mnt/MC1
/usr/bin/overviewer.py --config=/var/www/html/overviewer/ov.conf
```

ソースコード 22: cron 設定 (筆者環境)

```
50 0-23/4 * * * /var/www/html/overviewer/ov.sh >> /var/www/html/overviewer/mc_world/
  ov_log.txt 2>&1
```

10.3 ルータ設定例：BRT-AC828

ドメインは一つしか取っていませんがホストが複数台あるので、ポート転送でアクセスを振り分けます。以下の通り設定しています。

1. ログインしたら「WAN」を選択します (図 15a の①)。
2.「ポートフォワーディング」を選択します (図 15a の②)。
3.「ポートフォワーディングを有効にする」を ON にします (図 15a の③)。
4.「プロファイルを追加」を選択します (図 15a の④)。
5. カスタム設定を変更します (図 15b の枠内)。設定の値は表 5 を参照してください。

(a) 設定 1　　(b) 設定 2

図 15: ポート転送設定

サービス名	任意	
プロトコル	TCP	
外部ポート	80	<minecraft のポート>
内部ポート	外部ポートと同じ	
内部 IP アドレス	<http 用マシンの IP>	<本体用マシンの IP>

表 5: ポート転送設定

11 その他

11.1 SElinux について

Linux はしょっちゅう「アクセス権がありません (意訳)」というエラーが出ますが、大体の場合パーミッションか SELinux です。ここで「SELinux は disabled にしましょう！」とか言うと突然斧で脳天を唐竹割りされるのですが、説明しようと思うとそれだけで文量が 2 倍くらいになりそうなので流石に割愛します。SELinux の基本は「**まず Enforcing で動かしてみる、権限で怒られたら Permissive で動かしてみる、Permissive で動くなら SELinux のエラーを潰す**」です。調査方法については **setroubleshoot** の使い方を調べてみるとよいでしょう。筆者もよく SELinux で無限にキレ散らかしてるけど頑張ってね！

11.2 昔のマップが残る

本体をリセットしたけど Overviewer は使い回した際、旧マップが残って表示されたりします。`--forcerender` や`--check-tiles` である程度消えますが、遠いチャンクとかが少々残ったりして悲しいです。これについて Overviewer 公式ドキュメントの有り難い助言を確認しましょう。

公式ドキュメント：Docs » Frequently Asked Questions

You can run a render with –forcerender. This has the unfortunate side-effect of re-rendering everything and doing much more work than is necessary.
Manually navigate the tile directory hierarchy and manually delete tiles along the edge. Then run once again with –check-tiles to re-render the tiles you just deleted. This may not be as bad as it seems. Remember each zoom level divides the world into 4 quadrants: 0, 1, 2, and 3 are the upper left, upper right, lower left, and lower right. It shouldn’t be too hard to navigate it manually to find the parts of the map that need re-generating.
The third non-option is to not worry about it. The problem will fix itself if people explore near there, because that will force that part of the map to update.

英語で何言ってるかわかんねえって？俺もだガハハ！ものすげえざっくり言うと「`--forcerender` と`--checktiles` で頑張ってくれ！**まあでも誰かがそこでチャンク再生成すれば上書きされるから気にしないのも手だぞ！**」といった感じです。みんなも再生成まで諦めよう！筆者は諦めた！

11.3 MOD について

基本的に MOD のブロックは対応していません。ブロック処理が大元の python のソースコードにハードコーディングされているので、ソースを直で改修しないと追加することもできないようになっています。ただし、**MOD で生成されたディメンションをバニラブロックで表示できる範囲で出力する**ことは実は可能です。かなり虫食いにはなりますし、勘で指定することにはなりますが。

1. world データのディレクトリ構成を確認し、「DIM-<番号>」というディレクトリがあるのを確認します。多分これの内どれかが MOD で追加されたディメンションです。
2. config に renders を一つ追加し、**dimension** に「**DIM-1**」から順に入れてはレンダリングしてを繰り返します。その内ヒットします。がんばってください。

11.4 minecraft 最新 ver への追従について

本体のアップデートに対して Overviewer 側のアップデートは正直遅いです。特に正式版は中々出てこなく、追加されたブロックが描写できないパターンも多いです。 ただ開発自体は進んでおり、Overviewer の github(`https://github.com/overviewer/Minecraft-Overviewer`) で issues を探すとバージョン毎のブロックを今作っています！という記事が見つかります[14)]。そういった場合、ソースコードには既に反映されていることが多いので、章の手順を参考に最新のソースコードからビルドしてみると表示できたりします。DeepL 翻訳などを片手に github を頑張って探してみてください。

14） https://github.com/overviewer/Minecraft-Overviewer/issues/2046 など。

あとがき

まさか個人で一冊出すことになるとは……　何はともあれとりあえず一区切りできるレベルで間に合ってよかったです。現在 12/17 の 10 時過ぎ。前回[15]は 12/26 に後書き書いてるし今回は大成功だなガハハ！

前々から身内向けに minecraft サーバを公開していて、Overviewer に散々苦しめられた上に日本語資料が全然見当たらないので「もしかしてこれはブルーオーシャンなのでは？」とか勘違いした結果こんなにバカみたいな量の文章になりました。Overviewer の日本語資料がとにかく少ない上に、公式ドキュメント (英語) もよくよく読むと全然頼りないので、**機械翻訳した公式 doc で何となく結果を推測⇒実際に試す⇒出てきた結果から効果を逆算して説明文を起こす**をひたすら繰り返してました。genpoi 周りは地獄だった。

編集中も LaTeX[16] くんがなんか知らんけど subsection で undefined 吐いたりしてお前絶対そんなことないやろって言いながら再定義して誤魔化したりソースコードが真っ黒になったりページの調整で何もわからんになったりしましたが今となってはいいおもいでです。

あ、そうだ。その内無料公開しようとは思ってるんですが、取り急ぎ今ここまで読んでくれた人向け[17]に pdf 公開します。コマンドのコピペとかで電子あった方が便利でしょきっと。以下の url からこの文書を DL してみんなも Overviewer を使ってみよう！

`http://hiruuki.ddo.jp/c101_MC.pdf`

というわけでお疲れ様でした。よいお年を。

発行	#奢りません
発行日	2022 年 12 月 30 日予定
印刷	ACCEA 様に依頼予定
連絡先	`https://twitter.com/hiruuki_plus`
本拠地	`https://hiruuki.hatenadiary.org/`
フェリーラ鯖	入りたい方は適宜ご相談ください。

15） C97 で変な合同誌を出してた。
16） ノルマ達成
17） 立ち読みで url 暗記した人とかでもまあいいや。せっかくだし Overviewer 使ってみてくれよな。