2016
秋号

百萬石
Hyakumangoku Vol.47

Published by MMM

部長挨拶

f_dita

f_dita@mma.club.uec.ac.jp

「百萬石」をお手に取っていただきありがとうございます。弊誌 (以下、部誌) は電気通信大学の公認サークル MMA が新歓期 (春号) と弊学が開催する調布祭時期 (秋号) の年 2 回発行・頒布しているものです。MMA とは"Microcomputer Making Association"の頭文字に由来しており、1975 年に創立されたサークルです。現在では計算機に関連することを中心にネットワークやセキュリティなどの分野でも活動しています。部誌には MMA の部員が各々に行なっている活動を記事にしたものが掲載されています。内容は多岐に渡り高度なプログラミングの話から旅行記など MMA の活動の自由さが感じられるものとなっております。

早いことで、私が部長に就任して 10 ヶ月が経ちました。本年度では、MMA の 40 周年記念パーティーがあったり、夏コミで部誌を販売したりとさまざまな行事がありました。もう数ヶ月で任期も終わり、後輩に部を託し研究に専念することとなります。主要メンバーとして最後の調布祭ということで張り切っていく所存です。また、残りの任期は後輩育成のための勉強会等を取り行なっていきたいです。MMA の今後の活動に是非注目ください。

それでは、ジャンク市でのお買い物をお楽しみください。夏コミで販売を行なって部誌夏号の販売も行なっていますので、是非お求めください。

2016 年 11 月吉日

目次

第 1 章

Raspberry Pi で電子工作

f_dita

概要
Raspberry Pi 3 Model B を使ってステッピングモータを動かすプログラムを Python で製作します。

1.1 Raspberry Pi とは

本誌を読む皆さんには不要かと思いますが、簡単に説明させて頂きます。

図 1.1 Raspberry Pi 3 Model B

Raspberry Pi は、イギリスにある Raspberry Pi Foundation によって製造されています。ARM プロセッサを搭載したシングルボードコンピュータで、様々なモデルが販売されています。現在 (2016 年 11 月 25 日) で最新のモデルは Raspberry Pi 3 Model B で CPU がクアッドコアとなり、シリーズで初めて無線ネットワークデバイスが搭載されました。

Raspberry Pi は元々教育用デバイスとして開発されており、プログラミングや電子工作を簡単に始めることができます。Raspbian と呼ばれる Debian ベースの専用 OS の他にも Arch Linux や Fedora を使う事が可能です。Raspbian では標準で Python や Ruby がインストールされており、すぐにプログラミングを始めることができます。さらに、数式処理ソフトである、Mathematica が標準インストールされていて無償で使用することができます。電子工作についても、Raspberry Pi ボードに実装されている GPIO ピンやオプションで追加できるカメラモジュールを使用できます。これらのピンやカメラ

モジュールを扱える専用ライブラリが Python や Mathematica などに用意されていて簡単に利用できます。

1.2　ステッピングモータとは

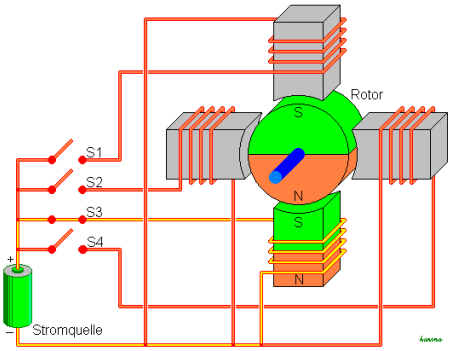


図 1.2　ステッピングモータの構造

ステッピングモータとは、パルス電力を入力することで、それに同期して動作する同期電動機です。回転量がパルスの数に比例しているので、簡単な回路で正確な位置決め制御が実現できます。デジタル制御回路との相性がよく、フィードバック回路が必要ないので簡単に扱うことがきます。ただし、エネルギー効率が悪く、高周波のパスルを入力すると脱調を起こします。脱調とはパスル周波数が高すぎて、パルスと回転の同期が外れる状態のことをいいます。

1.3　製作

いよいよ本題に入ります。

1.3.1　構成

今回使用した部品は以下のものです。

- Raspberry Pi 3 Model B
- ST-42BYG020 (ステッピングモータ)
- TD62003APG (トランジスタアレイ)
- 抵抗・電源・その他

Raspberry Pi の GPIO ピンの出力ではモータを駆動させるだけの電力が出せないので TD62003APG で増幅を行ないます。

ステッピングモータを駆動させるためには先述したようにパルス電圧を入力させる必要があります。今回使用するステッピングモータはユニポーラ方式なので次のような構成を考えました。

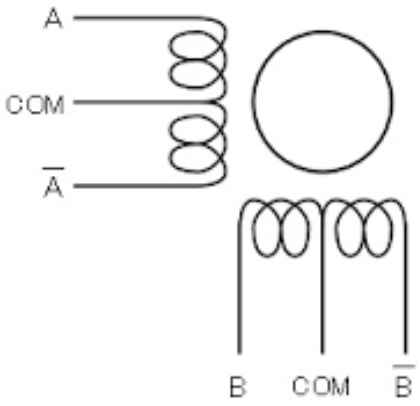


図 1.3　ユニポーラの構造

まず、COM の端子をそれぞれグランドに接続します。次に、A 端子だけに電圧をかけると順方向に、$\overline{A}$ 端子だけに電圧をかけると逆方向に電流が流れます。これでパスル電圧を作ることができました。電圧の ON/OFF を Raspberry Pi で制御すれば任意の周波数のパスル電圧を生成できます。これをもう一方の B,$\overline{B}$ 端子の組にも適応します。つまり、Raspberry Pi の GPIO ピンを 4 つ、トランジスタアレイのチャンネルも 4 つ使用することで、2 つのパルス電圧を作ることができます。2 つのパルス電圧を入力できるので、そのタイミングをずらすことでステッピングモータの正回転と逆回転を操作できます。

1.3.2　回路設計

構成での考えをもとに回路を作っていきます。GPIO ピンはどれを使ってもいいのですが、今回は GPIO の 7,8,9,10 を使用します。ピン番号はそれぞれ 26,24,21,19 ピンです。図 1.4 に実際に使った回路

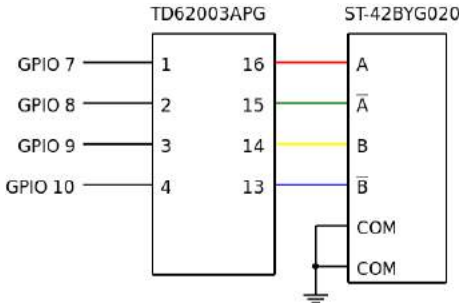


図 1.4　回路構成

を載せます。回路図には描いていませんが、IC の電源や抵抗等も忘れないようにしてください。

1.3.3　プログラム

それでは最後にプログラムを作成していきましょう。今回使用する言語は Python です。GPIO 用の専用のライブラリを使用するので、その説明を少しします。Python に関する詳しい説明は省略させて頂きます。

まずはライブラリのインポートです。

```
>>> import RPi.GPIO as GPIO
```

次に使用するピンのセットアップを行ないます。

```
>>> GPIO.setmode(GPIO.BCM)
>>> GPIO.setup(7,GPIO.OUT)
```

この操作によりまず 1 行目で GPIO ピンの指定方法を決めています。2 行目で GPIO の 7 番ピンを出力ピンに設定しています。これで、7 番ピンの設定が完了しました。

次に、7 番ピンを実際に操作してみましょう。7 番ピンは出力ピンなので、出力を ON/OFF にする操作ができます。設定した初期状態では出力は OFF なので ON にしてみます。

```
>>> GPIO.output(7,GPIO.HIGH)
```

これで出力が ON になりました。OFF にする場合は同様に

```
>>> GPIO.output(7,GPIO.LOW)
```

で簡単に OFF にできます。

GPIO ピンの使用が終了したら、GPIO ピンをリセットしましょう。リセットするには次の操作を行なってください。

```
>>> GPIO.cleanup()
```

以上でライブラリの説明を終わります。この説明は今回使用するものしか取り扱っていないので、興味がある方は自分で調べてみてください。

これの操作を使い出力の ON/OFF を繰り返せばパルス電圧を発生させることができます。

それでは実際に動作させたコードを載せます。

```
import RPi.GPIO as GPIO
import time

# GPIO setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(7,GPIO.OUT)
GPIO.setup(8,GPIO.OUT)
GPIO.setup(9,GPIO.OUT)
GPIO.setup(10,GPIO.OUT)

# 周波数の調整
sleeptime = 0.01
```

```
# 90度回転
for i in range(0,13):
  GPIO.output(7,GPIO.HIGH)
  GPIO.output(8,GPIO.LOW)
  time.sleep(sleeptime)
  GPIO.output(9,GPIO.HIGH)
  GPIO.output(10,GPIO.LOW)
  time.sleep(sleeptime)
  GPIO.output(7,GPIO.LOW)
  GPIO.output(8,GPIO.HIGH)
  time.sleep(sleeptime)
  GPIO.output(9,GPIO.LOW)
  GPIO.output(10,GPIO.HIGH)
  time.sleep(sleeptime)

# GPIO clean
GPIO.cleanup()
```

このコードではパルス電圧の周波数を調整するために time.sleep を使っています。また、ステッピングモータへのパルスの与え方には種類があり、今回は二相励磁モードを用いました。ステッピングモータは 1 パルスで動く角度 (ステップ角) が製品によって決っているので、そこから目標の角度までに必要なパルス数を計算します。今回使用したステッピングモータはステップ角が約 1.8 度なので 50 パルスで 90 度回転します。このコードでは 1 回 for 文が回るごとに 4 パルス発生するので、全部で 52 パルス発生しています。なので正確には 90 度ではなく、93.6 度回転していることになりますが、今回はそんな細かいことは無視します。

1.4　まとめ

このようにして、Raspberry Pi でステッピングモータを操作することができました。先程のコードを少しいじれば簡単に逆回転するコードを作ることができます。また sleeptime の値を変化させればステッピングモータの回転速度を変化させることができます。for 文の回数を調整すれば回転角度を自由に調整できます。このように非常に簡単なコードでステッピングモータを自由に操作することが可能です。また、ステッピングモータ以外にもスピーカーや LED などの操作の同じように簡単にできます。さらに今回は使用しませんでしたが、GPIO ピンは入力を受け付けることもできるので、スイッチやセンサからの値を取得して動作するプログラムを製作するできます。

第 2 章

if 式とユニットの重要性

mizunashi_mana

概要
プログラミング言語によって、if は文であったり式であったりします。if が式であることはそれだけで非常に利便性を向上させます。しかしながら、多くの標準的な言語、C++/Java/JavaScript/Python といった言語では if が式ではないため、この利便性はあまり理解されていないようです。 今回は、if 式がどういうように役に立つのか、そして if 式を実現させるために必要なユニット型について説明したいと思います。

2.1　if 式とは

2.1.1　if 式とはどういうものか

if 式とは、多くの言語に存在する式となれる分岐構文のことです。if 文は多くの言語で見られる基本的な構文です。if 式はその構文をさらに拡張したものです。一部の言語では制御構造ではなくただ単なる組み込みの演算子として扱われています。

2.1.2　動的型付け言語の if 式

- Ruby の if 式
  Ruby はブロックという、複数の文を集めた実行単位を持ちます。Ruby の if 式は単純に、一つの条件式と二つのブロックを受け取って新たな文を作るものになります。ブロックは、最後に評価された値を返します。if 式は実行したブロックの返り値をそのまま返します。if のみで else が無い場合、条件が偽の場合 nil が返ってきます。

```
i = if true
  42
else
  0
end
```

- CoffeeScript の if 式

CoffeeScript も Ruby と同じく、複合文においては最後に評価された値を返します。if 式は実行した複合文の値をそのまま返します。if のみの場合、条件が偽の時は undefined が返るようになっています。

```
i = if true
  42
else
  0
```

- Clojure の if 式

  Clojure などの Lisp 系では、cond と if どちらかしかない言語、どちらかがどちらかを基にしたマクロになっている場合など、様々なケースが存在しますが、if がある言語では一様に if 式は一つの条件部の S 式と二つの S 式をとり、条件に合わせて S 式の結果を返す式になります。Clojure では加えて if だけの場合は条件が偽の場合 nil が返る仕様になっています。

```
(def i
  (if true
    42
    0))
```

2.1.3 静的型付け言語の if 式

- Rust の if 式

  Rust では複合文はそれ自体が返り値を持ちます。if は単純に一つの条件式と二つの複合文を受け取り条件に合わせて複合文の結果を返します。複合文において if と else のブロックの返り値の型は同じである必要があります。else が無い場合、返り値はユニットになります。

```
let i: i32 = if true {
  42
} else {
  0
};
```

- Scala の if 式

  Scala も Rust と基本的には同じです。ただし、Scala では else の無い if 式は基本的には Any 型を持ちます。そのため、条件が真の場合ブロックの返り値を、偽の場合ユニットを持つ Any 型の値が生成されます。

```
val i: Int = if (true) {
  42
} else {
  0
}
```

- Kotlin の if 式
  こちらもほぼ Rust と同じです。しかし、Kotlin では else の無い if は式ではありません。

```
val a: Int = if true {
  42
} else {
  0
}
```

- Haskell の if 式
  Haskell は全てが基本的には式であり、複合文は許容されていません。そのため、if 文は非常に単純で、条件式一つと式二つをとり条件に合わせて式を評価します。else の無い if 式は存在せず、代わりにデータ構造によってそのようなものと等価な関数 (例えば、モナド上の when など) が用意されています。

```
i :: Int
i = if True
  then 42
  else 0
```

- StandardML の if 式
  StandardML も Haskell とほぼ変わりません。

```
val i = if true
  then 42
  else 0
```

2.2　if 式を導入するには

2.2.1　全ての文は式

Rust では一部の文を除き全てが式になっています。if や match をはじめ、loop、for や while、ブロックでさえ式になっています。想像しにくいかもしれませんので、次のプログラムを見てみてください。

```
// an if expression
println!("{:?}", if true { 1; }); // -> ()

// an if-else expression
println!("{:?}", if true { 1 } else { 2 }); // -> 1

// a match expression
println!("{:?}", match 1 { 0 => false, _ => true, }); // -> true

// a loop expression
println!("{:?}", loop { break }); // -> ()

// a for expression
println!("{:?}", for _ in 0..1 { break }); // -> ()

// a while expression
println!("{:?}", while false { break }); // -> ()

// a block expression
println!("{:?}", { 1 }); // -> 1
```

() はユニットという値を表します。Rust ではこのようにユニットを使うことで、うまく全てを式にしていて、ブロックに値を持たせる方法を作成することで、書き方に一貫性と簡潔性を持たせるようにしています。しかしながら、これらは理論的な一貫性においては綺麗ですが、実用ではそれほど役に立つのか、と思われる方もいるでしょう。今回の例ではそれほど役に立っているようには見えません。if-else 式や match 式はまだ用途が理解できますが、for 式や while 式などは一体なんの役に立つのでしょうか？

例えば、Rust では多くの場面でマクロが併用されています。println!もマクロの一つです。マクロには様々なものがありますが、例えば C で複雑なプログラムを書く際、define は非常に力不足でした。例えば define に for 文を渡すなどといったことはできませんでしたし、非常に複雑な構文を define で定義させることについても様々な制約がありました。しかしながら、Rust のマクロはそれらを反面教師に非常に広範囲に適用可能なマクロを提供しています。その広範囲の適用に、制御文が式であることが一役買っています。制御文が式であるということは、普通の式と制御文を区別しなくて良いということです。これによって式全般向けに作られたマクロを制御文に適用させることもできますし、その逆、つまり制御文用に作られたマクロに式をなんなく適用することも可能なのです。

他にも大きな利点がありますが、それについては後ほど紹介しましょう。さて、今回はこれらの制御文のうち if 文に限定して利点を観察してみます。if 式は単純に全ての言語がなんなく採用できるものではありません。まずは、if 式の採用に重要な二つの要素を見ていきましょう。

2.2.2　副作用と if 文

if 文を式にするためのキーワードの一つは副作用です。if 文は副作用の作用点が明確になっている必要があります。例えば次のコードを見てください。

```
fn ifelse<T>(condition: bool, ifv: T, elsev: T) -> T {
  match condition {
    true => ifv,
    _    => elsev,
  }
}

if 1 == 1 {
  println!("{}", 1);
} else {
  println!("{}", 2);
}

ifelse(1 == 1, {
  println!("{}", 1);
}, {
  println!("{}", 2);
});
```

ifelse 関数は、if 式を模した関数です。しかしながら、if 式と ifelse 関数の使用例では、出力が異なります。関数呼び出しでは、全ての引数が評価されます。その後に ifelse 関数内部での match 式部分が評価されるのです。if 式では、片方のブロック部しか評価されません。さらに、条件部の評価を終えてから実際の文が評価されるという制約もつきます。

これは、当たり前のことに見えますが、実装上とても重要なことです。このため、あなたが if 式と同義の関数などを実装する場合、これらの制約について注意する必要があるのです。多くの言語では、正格評価が基本であり、ブロックを遅延させるようなサポートはあまり搭載されていません。このため、if 式と同等の関数を実装するためには、引数に無引数をとるクロージャを渡すようにし、クロージャによって副作用の作用点をクロージャ実行時に遅延させるといった実装をする必要があります。

条件分岐が処理の評価の制御においてもとても基礎的な部分になっていることは、昔から広く知られており、三項条件演算子や論理和演算子、論理積演算子などは if 文と同じく、処理を遅延させるサポートがなされています。これによって私たちは if 式がない言語でも基本的な式の評価においてパフォーマンス上の恩恵を受けることができます。しかしながら、複雑な処理の分岐が必要な場合も様々なケースで存

在します。これらにおいて遅延のためのクロージャを用いることは、実行速度、使用メモリの両方の点で負荷となります。if 式は、副作用の制御という点でとても有用な部品です。

ところで、もちろん if 式自体の副作用の作用点も、式の一つとして明確になっている必要があります。これは、他の式と同じように、if 式が実行される際となるでしょう。これらの事項はとても基本的なことに聞こえますが、言語実装者にとってはきちんと確認しておくべき事柄です。

2.2.3　ユニットとは

もう一つ、if 式を採用するのに重要になってくるのが、ユニットという概念です。以下のプログラムを見てください。

```
fn print_i32(i: i32) -> () {
  println!("{}", i)
}
```

これは、Rust では以下のものと同じです。

```
fn print_i32(i: i32) {
  println!("{}", i)
}

fn print_i32(i: i32) {
  println!("{}", i);
}

fn print_i32(i: i32) {
  println!("{}", i);
  return ()
}
```

println!関数は関数の内容を実行し終えるとユニットという値を返します。ユニットとはその名の通り、たった一つの値しか持たないユニット型（単一型）の値のことです。Rust では返り値の型を省略した場合、ユニット型を返す関数になります。これが if 式の採用に一役買っています。

if 式はそれぞれの分岐の結果を返すのでした。では、例えば片方で Hello、もう片方で World と出力するような if 式はどのような結果を返せばいいのでしょうか？式とは計算すると必ず値が返ってくるものです。このような場面でユニットが重要になってくるのです。例えば Generics が使われている場合などは、上記のような場合 if 式に特別な意味を与えることはできません。しかし、ユニットを使うことで、単純に if 式においてそういうものを許容することができるのです。

```
if flag {
  println!("Hello");
} else {
  println!("World");
}
// -> ()
```

ところで、void とユニットの違いはなんでしょう？多くの言語では void という何も値が無いことを示す型が用意されています。ユニット型は名前は違えど、この型となんら違いが無いように思えます。よく思い出してほしいのですが、void が採用されている言語では if は式になっていません。これは別にその言語が歴史的事情で if 式を導入していないのではなく、void とユニットには明確な違いがあり、それこそが if 式が導入できない理由になっているからです。

その違いは非常に単純で、型に含まれている値が存在するかしないかです。void 型をもつような値は存在しません。void は本当に値を持たない型で、修飾のため付けられる場合が多いわけです。それに対して、ユニット型はたった一つ値を持つ型です。通常 () という値が所属する型のことを指しますが、Rust ではユニット型は作り放題です。

```
struct UnitLike {} // ~= ()
struct UnitLike2; // ~= ()

let x = UnitLike {};
let y = UnitLike2;
```

それぞれは別の型で、性質としては同型といって本質的には同じですが、別々の型を表します。() は標準が提供している型で、() という値をたった一つ持つことになります。ユニットは非常に広範囲に応用がききますが、それについては今回は省略しておきます。

この概念の違いに最初は戸惑うかもしれません。しかしながら void に比べてユニットはそれほど遜色なく使用できるもので、しかも void より応用範囲が広いのです。ではなぜユニットを採用しないのでしょうか？これは、オブジェクト指向の言語では少々型安全な実装を行うことが難しいという側面があるからです。また既存の言語の場合、互換性を著しく破壊するため採用が見送られているケースも多いでしょう。しかしながら、最近の言語では実用性が評価されてきています。Rust や Kotlin などでは void ではなくユニット型が採用されています。ユニットのインスタンスが毎回関数呼び出し度に生成され効率が悪いのでは無いかと思う方もいるかもしれません。しかしながら、ユニットは値が必ず一つしか無いため、静的解析が容易であり、コンパイル時にユニットの値領域を確保する処理に変換せず、別の等価な文を生成でき、一切パフォーマンスに影響しないようにすることも可能です。

2.3　if 式が役に立つ場面

2.3.1　三項条件演算子の上位互換

if 式を導入していない言語の多くには、三項条件演算子が存在します。例えば C 言語では次のような使い方ができます。

```
const int i = true ? 42 : 0;
```

if 式はこの演算子の完全な上位互換です。if 式はこの演算子にはない以下の要素を持っています。

- 読みやすくネストしやすい
  これは非常に重要です。以下の C と Rust のコードを見比べてみてください。

```
int n = i == 1
  ? 0
  : i == 2
  ? 1
  : i >= 3
  ? 2
  : -1
  ;
```

```
let n = if i == 1 {
  0
} else if i == 2 {
  1
} else if i >= 3 {
  2
} else {
  -1
}
```

  どちらも同じ行数しか消費していません。にも関わらず Rust の方が断然読みやすいことにあなたは気づくでしょう。三項条件演算子は特に分岐の中で分岐を書く場合非常に構造が分かりにくくなります。そして、IDE などのフォーマッタのサポートも受けにくいのです。
- 文を扱える
  三項条件演算子は、その性質上式しか扱えません。if 式の場合、言語によっては一貫性のために式しかとれずブロックも式となっている言語も存在しますが、共通して言えることは複合文が扱え

るような仕様になっているということです。C では以下のようなことは不可能です。

```
int n = if i == 1 {
  int a = 0;
  for (const j = 0; j < 10; j++) {
    a += o[j];
  }
  a
} else {
  0
}
```

しかし、if 式を持つ言語では総じて可能です。

2.3.2　イミュータブルとの連携

if 式が大きな役割を担うのがイミュータブル変数との連携です。イミュータブル変数は、プログラミングスタイルを大きく改善することが知られています。しかしながら、if 式を採用していない多くの言語では、その言語の制約上本来なら変数をイミュータブルにできる箇所が、できないがためにミュータブルになってしまうといったことがあります。例えば、次のような例です。

```
int i;
if (flag) {
  int n = 0;
  for (int j = 0; j < N; j++) {
    n += j;
  }
  i = n;
} else {
  i = 0;
}

printf("%d\n", i);
```

この例では、変数 i は初期以降変更されないため、const 修飾できるはずです。例えば、Rust では次のようにイミュータブルに初期化ができます。

```
let i = if (flag) {
  let mut n = 0;
  for j in 0..N {
    n += j
  }
  n
} else {
  0
};

println!("{}", i);
```

これによって、変数 i はこれ以降アクセスのみが出来ることになります。この初期化法は単純にイミュータブルでなくても、利があります。次の初期化を見比べてみてください。

```
let mut n1;
if flag {
  n1 = 1;
  for j in 0..N {
    n1 += j
  }
} else {
  n1 = 0;
}

let mut n2 = if flag {
  let mut n = 1;
  for j in 0..N {
    n += j
  }
  n
} else {
  0
};
```

一番目の例が if 式を使わない例、二番目が if 式と初期化を組み合わせた例です。どちらもそれほど変わりませんが、一番目の例はいくつか注意が必要な場所があります。例えば、一番目の例の場合、あなたは else 区で初期化を忘れてしまったとしても素通りしてしまうだろうということです。そして、もう一

つは if 区の中で初期化する変数の値が順に変わっていくわけですが、この初期化手順がターゲットとなる変数と独立していないため、例えばデバッグ時 if 区の中のデバッグとターゲット変数が容易に切り離せない点です。そして、最後にプログラムを見た時どちらが意味を見て取りやすいかです。通常変数が何も初期化されていない状態で次に if 文が来た場合次が初期化と考えるケースは多いでしょう。しかしながら幾つかの場合、if 文で分岐を行う前に共通の特殊な処理をしなければいけないなどの制約がつくこともあり、パッと初期化処理がどこまでなのかはわかりづらい場合もあります。

ブロックと if 式をうまく組み合わせれば上の三つの問題はすべて解決できます。これは if 式を採用する利点として十分でしょう。

2.3.3　分岐と結果の制御

プログラミングをする時、結果を分岐したいことは多々あるでしょう。if は分岐の基本的な制御機構です。しかし、if 文しか無い場合結果の分岐制御は不十分な場合があります。以下の C のコードを見てください。

```
int res;
if (flag) {
  res = 1;
} else {
  res = 2;
}
func(res);
```

これはよくあるコードの一例です。しかしながら、多くの余分な文が含まれています。今回の場合は次のように書けます。

```
func(flag ? 1 : 2);
```

しかしながら、上記で述べた通り三項演算子は文を扱えないという難点があります。プログラムが複雑になるにつれ、複雑な処理に分岐した結果を関数に引き渡したい時は多数存在します。そのような場合に if 式は強力な力を発揮します。

```
func(if flag {
  // anything
  result
} else {
  // something
  result
});
```

また、分岐する関数の結果を返す場合も if 式は非常に有益です。

```
fn factorial(n: i32) -> i32 {
  assert!(n >= 0);

  fn go(n: i32, cont: i32) -> i32 {
    if n == 0 {
      cont
    } else {
      go(n - 1, cont * n)
    }
  }

  go(n, 1)
}
```

あなたは、特に return などに気にせずほとんど最後の部分だけを追うことで、プログラムの全体像をつかむことができます。これによってプログラムを簡潔にすることが可能なのです。

2.4　if 式の重要性

2.4.1　依存を減らすための if 式

if 式は、これまで見てきたように様々な場面で役にたつことを紹介してきました。さて、これらの if 式の活用例は、加えてプログラミングスタイルに大きな効能をもたらしてくれます。それは無駄な依存を減らしてくれるというものです。

例えば、if 式はイミュータブルと相性が良いという話をしました。あなたはこれによって、無駄に変数をミュータブルにする必要が無いことに加えて幾つかの効用も得られるのです。例えば、無理やりイミュータブルにするために初期化用の関数を作らなくても自然に分岐的な初期化を書けるようになります。もし汎用的な初期化や様々な条件分岐が必要なとても複雑な初期化が必要な場合初期化関数を実装したほうがいいでしょう。しかし、一行二行のその場でしか使わない関数はあまり定義する必要が無いものです。if 式は不必要な変数を完全に定義しなくて良いという点で画期的です。

また、if 式やブロックを式にすることによって、無駄な変数を作成するのを防ぎ、またコントロールフローを明確にすることも可能です。複雑なコードは得てしてその場限りのものが多いでしょう。もしブロックや if 式が式でなかった場合、もし結果をなんらかの場所に格納しなければならなかった場合、結果用の変数を作成する必要が出てきます。しかし例えば、ブロック内の処理によって処理を値に何度も施し最終的な値を返すような処理がしたい場合、私たちは変数を流用しがちです。これによって結果用の変数が、処理内部でも使用され、プログラムが必要以上に複雑なフローになりがちです。もし、結果用の

変数をその後も使い回すと、バグの温床になりやすく、どこでその変数を使用しているかの依存があまりはっきりしません。

if 式は結果用の変数を必要としないため、必要に応じて変数の作成が行え、また if 式内部での処理を独立させることが可能です。私たちは if 式がどのような値を返すかは if 式のブロックの最後の部分に着目すればよく、関数の引数に分岐した処理を行った結果を渡したい場合も不必要な依存を作らなくても良くなります。これは不必要な部分が少なくなるとともに、それぞれのフローを独立させることを容易にし、プログラムを細かい単位で着目しやすく、またアルゴリズムの本質を捉えるのに非常に有益なことです。

関数の返り値に if 式を使用することは、結果用の変数の依存をなくすだけでなく、プログラムを簡潔にします。つまり、それぞれの分岐に対して return を明記しなくても良くなるため、どこで返り値が return で指定されているかに感覚を研ぎ澄ませる必要がなくなるのです。また、これによって自然に return を使った複雑なフローを用いなくてもプログラムが書けるようになります。

2.4.2　最後に

if 式はプログラミングスタイルを劇的に改善してくれる、基本的で素朴なアイディアです。そして、このアイディアから学ぶことは多いでしょう。このアイディアがより広く浸透してくれることを願っています。

第 3 章

レイトレーシング

su_zu

概要
レイトレーシングというグラフィック技術を通して BMP の仕様, cuda, C++14 の constexpr について紹介したいと思います.

3.1 BMP

BMP, Windows bitmap は Windows が標準で使用している画像ファイル形式である. 比較的無圧縮であることが多いので今回は BMP に出力することにする。

3.1.1 仕様

基本的に以下より記述するバイトオーダはリトルエンディアン形式である. リトルエンディアンとは下位から上位に順番に値を格納する方式である.

ファイルヘッダ

要素	バイト数
ファイルタイプ	2
ファイルサイズ	4
予約 1	2
予約 2	2
イメージデータオフセット	4

ファイルタイプ
BMP であることを示す."BM"が値として格納される.

ファイルサイズ
BMP ファイルのバイト単位でのサイズを符号無し整数値で入れるが厳密でないエンコーダはこの

値を 0 を格納してもよい. 今回は 0 を格納する実装にする

予約 1

将来の拡張領域

予約 1

将来の拡張領域

イメージデータオフセット

ファイルヘッダの先頭アドレスからビットマップデータの先頭アドレスまでのオフセット

情報ヘッダ

情報ヘッダは様々な仕様が現在あるが最も一般的な INFO タイプを紹介する.

要素	バイト数
ヘッダサイズ	4
幅	4
高さ	4
プレーン数	2
ビット数 per ピクセル	2
圧縮タイプ	4
イメージデータサイズ	4
水平解像度	4
垂直解像度	4
カラーインデックス数	4
重要インデックス数	4

ヘッダサイズ

情報ヘッダのデータサイズが符号無し整数値で入る.INFO タイプでは必ず 40 となります.

width size

ピクセル単位での width が格納される.

height size

ピクセル単位での height が格納される.

プレーン数

プレーン数が符号無しで格納される.BMP では常に 1 となる.

ビット数 per ピクセル

1 ピクセルを表すのに必要なビット数が符号無し整数値で格納される. 今回は 24 を用いる.

圧縮タイプ

圧縮形式を表す符号無し整数値が格納される. 今回は無圧縮なので 0 を入れる.

イメージデータサイズ

イメージデータのバイト単位でのサイズが符号無し整数値で格納される.

水平解像度

水平方向のピクセル数 per 1 メートルが符号無し整数値で入る.

垂直解像度

上と同様

カラーインデックス

カラーパレットの色数が符号整数値で入る

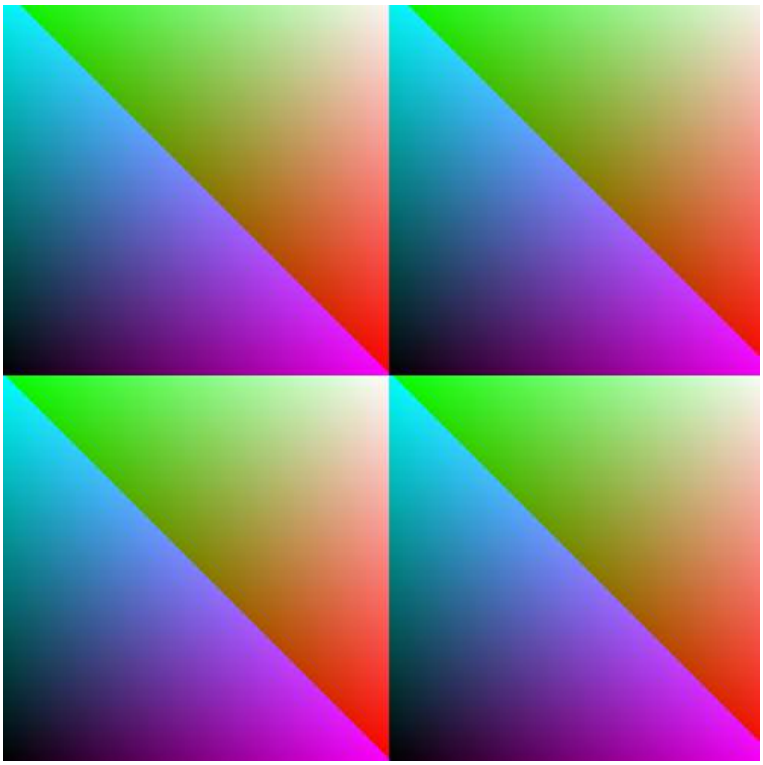
重要なカラーインデックス数

カラーパレットの色の中で正確に表示するべき重要な色のインデックスを符号無し整数値で入る

実装例

```
#include <fstream>
#define WIDTH_NUM 500
#define HEIGHT_NUM 500

void write2byteLittleEndian(unsigned short data, std::ofstream &bmp_file){
    bmp_file.write((const char *)&data, sizeof(char)*2);
}

void write4byteLittleEndian(unsigned long data, std::ofstream &bmp_file){
    bmp_file.write((const char *)&data, sizeof(char)*4);
}

int main(){
    std::ofstream bmp_file("output.bmp", std::ofstream::binary);

    // ファイルタイプ
    bmp_file.write((const char *)"BM", sizeof(char)*2);

    // ファイルサイズ
    write4byteLittleEndian(0, bmp_file);

    // 予約領域1
    write2byteLittleEndian(0, bmp_file);

    // 予約領域2
    write2byteLittleEndian(0, bmp_file);

    // イメージデータオフセット
    write4byteLittleEndian(14+40, bmp_file);

    // ヘッダサイズ
    write4byteLittleEndian(40, bmp_file);

    // 画像の幅
    write4byteLittleEndian(WIDTH_NUM, bmp_file);

    // 画像の高さ
    write4byteLittleEndian(HEIGHT_NUM, bmp_file);

    // プレーン数
    write2byteLittleEndian(1, bmp_file);
```

```
    // ビット数 per ピクセル
    write2byteLittleEndian(24, bmp_file);

    // 圧縮形式
    write4byteLittleEndian(0, bmp_file);

    // イメージデータサイズ
    write4byteLittleEndian(0, bmp_file);

    // 水平解像度
    write4byteLittleEndian(0, bmp_file);

    // 垂直解像度
    write4byteLittleEndian(0, bmp_file);

    // カラーインデックス
    write4byteLittleEndian(0, bmp_file);

    // 重要なカラーインデックス数
    write4byteLittleEndian(0, bmp_file);


    for(int h=0;h<HEIGHT_NUM;h++){
        for(int w=0;w<WIDTH_NUM;w++){
            unsigned char r = w % 255;
            unsigned char g = h % 255;
            unsigned char b = w + h % 255;

            bmp_file.write((const char *)&(b), sizeof(char)*1);
            bmp_file.write((const char *)&(g), sizeof(char)*1);
            bmp_file.write((const char *)&(r), sizeof(char)*1);
        }
    }

}
```



3.2　レイトレーシング

レイトレーシングとは ray(光) 追跡して主に 3 次元 CG をレンダリングする技法です．日本語では「光線追跡法」とも言われている．

今回は球をレンダリングすることを通じてレイトレーシングをご紹介する．

ソースコードには以下のような構造体を定義して使用しております．

```
// 三次元ベクトル
struct Vec3{
    double x;
    double y;
    double z;
};

// RGB
struct Rgb{
    unsigned char r;
    unsigned char g;
    unsigned char b;
};
```

3.2.1　球の交差判定

原点中心の半径 r の球は以下の式で表すことができる.

$$x^2 + y^2 + z^2 = r^2 \tag{3.1}$$

ray(光) をベクトル表すと

$$P(t) = Q + tV \tag{3.2}$$

x, y, z に代入すると

$$(Q_x + tV_x)^2 + (Q_y + tV_y)^2 + (Q_z + tV_z)^2 = r^2 \tag{3.3}$$

$$(V_x^2 + V_y^2 + V_z^2)t^2 + 2(Q_xV_x + Q_yV_y + Q_zV_z)t + Q_x^2 + Q_y^2 + Q_z^2 - r^2 = 0 \tag{3.4}$$

2 次多項式なので判別式を計算して交差判定を行う.

$$a = V^2 \tag{3.5}$$

$$b = 2(Q \cdot V) \tag{3.6}$$

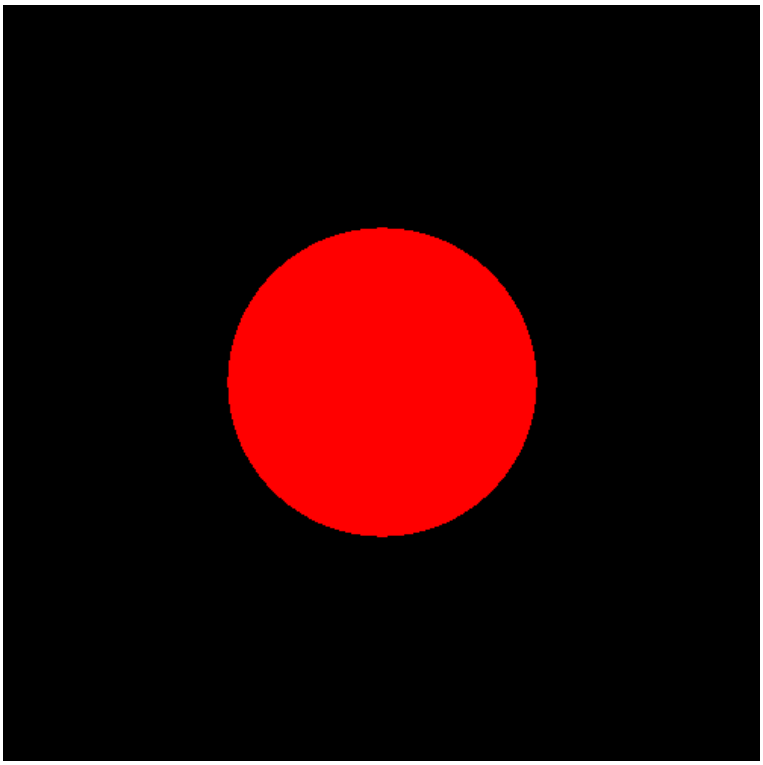
$$c = Q^2 - r^2 \tag{3.7}$$

$$D = b^2 - 4ac \tag{3.8}$$

$D >= 0$ なら ray は球に交差する.
2 点通る場合は始点 Q に近い t の値は以下のようになる

$$t = \frac{-b - \sqrt{D}}{2a} \tag{3.9}$$

```
// inner_prodは内積
double a = inner_prod(ray.direction, ray.direction);
double b = 2*inner_prod(ray.origin, ray.direction);
double c = inner_prod(ray.origin, ray.origin) - (sphere.r * sphere.r);
double d = b*b - 4*a*c;
if(d >= 0.0){
    rgb.r = sphere.rgb.r;
    rgb.g = sphere.rgb.g;
    rgb.b = sphere.rgb.b;
}
```



球を表示することはできましたがのっぺりとしていて奥行きなど今のままでは全く感じません．そこで３ライティングの処理を施して立体的に見えるようにします．

まず法線ベクトルを定義します．ray が球と交差するときの t は $t = \frac{-b-\sqrt{D}}{2a}$．球の法線ベクトル $n =$ (ray の交点) - (球の中心)

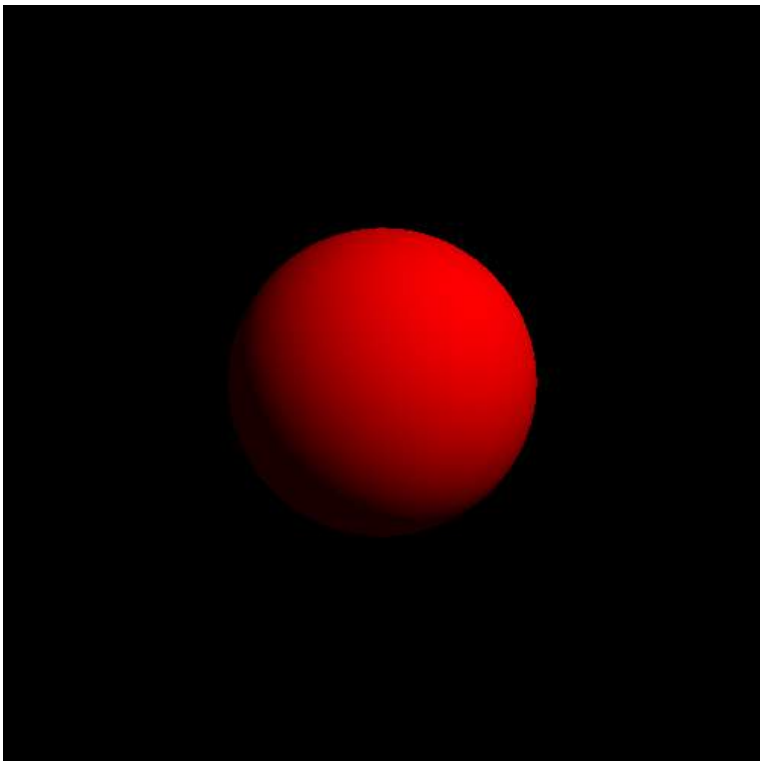
```
double t = (-b - std::sqrt(d))/ (2*a);
// addはベクトルの加法，prodはベクトルの乗算
Vec3 hitPoint = add(ray.origin, prod(ray.direction, t));
// normalizeは正規化
Vec3 normal = normalize(sub(hitPoint, sphere.position));
```

球に交差する光は拡散反射 (乱反射) します．これはランバート反射という反射モデルを用いて表すことができます．

L は面から光源のベクトル, N は法線ベクトル, C は色, I_L は入射光

$$I_D = L \cdot N * C * I_L \tag{3.10}$$

```
double d = clamp(inner_prod(normalize(Vec3{2.0, 2.0, 3.0}), normal), 0.1,
    1.0);
// clampは0.1から1.0に丸めている
rgb.r = sphere.rgb.r*d;
rgb.g = sphere.rgb.g*d;
rgb.b = sphere.rgb.b*d;
```



3.2.2　レイマーチング

先程の例で球を表示することはできたが球の関数を定義して ray の直線を代入して交差判定を行うのは少し手間ではないだろうか. そこでもっと簡単に球を表示してみる.

レイマーチングでは距離関数を定義してそのオブジェクトとの距離がおよそ 0 であれば交差すると判断する手法である. 距離関数とはオブジェクトと ray の距離を示す関数である.

例えば円であれば距離関数は以下のようになる

```
double distanceFunc(Vec3 rayPosition, Sphere sphere){
        return length(sub(rayPosition, sphere.position)) - sphere.r;
}
```

オブジェクトとの距離をループで距離関数を回して近づける

```
double distance = 0.0;
double rayLength = 0.0;
Vec3   rayPosition = ray.origin;

for(int i=0;i<64;i++){
    distance = distanceFunc(rayPosition, sphere);
    rayLength += distance;
    rayPosition = add(ray.origin, prod(ray.direction, rayLength));
}

if(std::abs(distance) < 0.1){
```

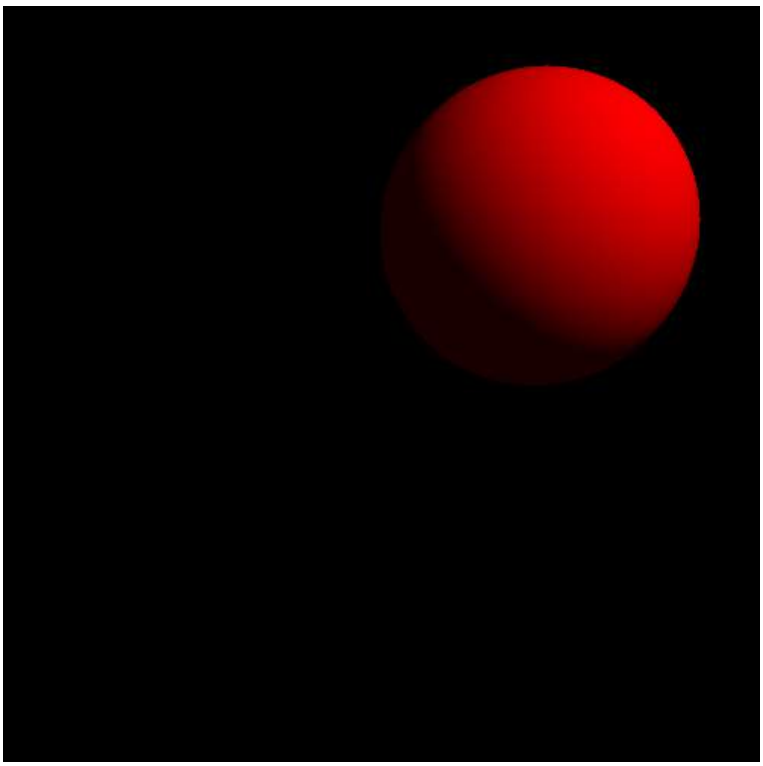
```
    rgb.r = sphere.rgb.r;
    rgb.g = sphere.rgb.g;
    rgb.b = sphere.rgb.b;
}
```

$f(x, y, z) = 0$ の曲面の法線 n は距離関数の勾配となる.

$$n = (\frac{\alpha f}{\alpha x}, \frac{\alpha f}{\alpha y}, \frac{\alpha f}{\alpha z}) \tag{3.11}$$

```
Vec3 n =  normalize(Vec3{
        distanceFunc(add(p, Vec3{d, 0, 0}), sphere) - distanceFunc(add(p,
            Vec3{-d, 0, 0}), sphere),
        distanceFunc(add(p, Vec3{0, d, 0}), sphere) - distanceFunc(add(p,
            Vec3{0, -d, 0}), sphere),
        distanceFunc(add(p, Vec3{0, 0, d}), sphere) - distanceFunc(add(p,
            Vec3{0, 0, -d}), sphere)});
```

```
Vec3 normal = getNormal(rayPosition, sphere);
double d = clamp(inner_prod(normalize(Vec3{2.0, 2.0, 3.0}), normal), 0.1,
    1.0);
rgb.r = sphere.rgb.r*d;
rgb.g = sphere.rgb.g*d;
rgb.b = sphere.rgb.b*d;
```

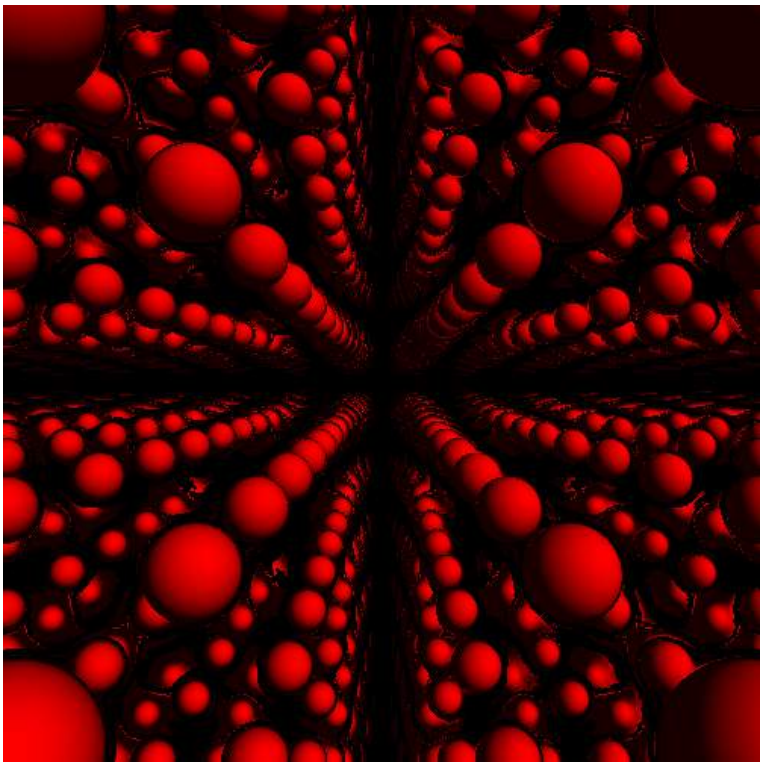


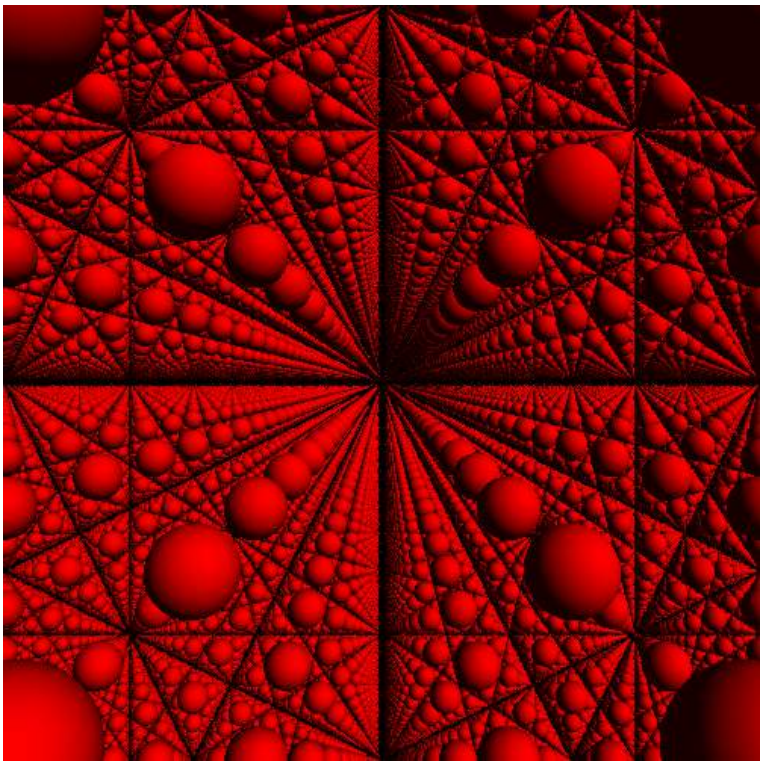
レイマーチングでは距離関数を以下のように変えると簡単にオブジェクトを繰り返すことができるようになる

```
double distanceFunc(Vec3 rayPosition, Sphere sphere){
    return length(sub(Vec3{
        rayPosition.x - 0.4*floor(rayPosition.x/0.4),
        rayPosition.y - 0.4*floor(rayPosition.y/0.4),
        rayPosition.z - 0.4*floor(rayPosition.z/0.4)}, Vec3{0.2, 0.2, 0.2}))
            - 0.05;
}
```

ベクトルの範囲を $-0.2 \sim 0.199999$ になるようにしている

これは距離関数をループで何回回したかで結果が変わってくる





3.3 GPU レイトレーシング

先程示したレイマーチングだと全ピクセルに対して 16 256 回計算しているので非常に時間がかかる.

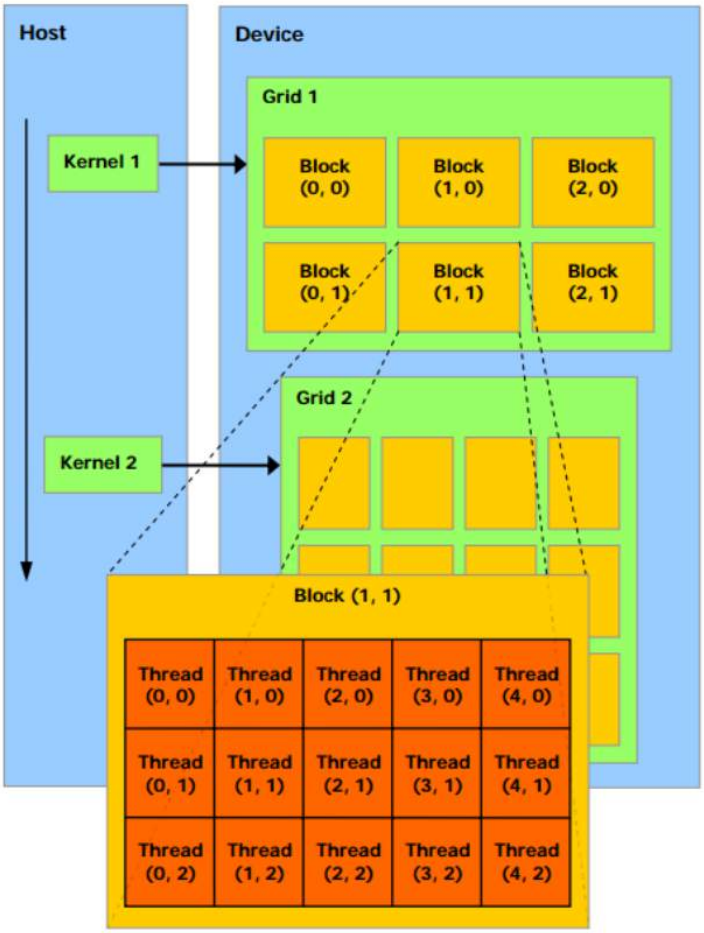
ここでは GPU を使ってレイトレーシングを行うことを試してみる. レイトレーシングではピクセルごとの処理は他のピクセルの結果に依存しないので並列に計算することができるので GPU と相性が非常に良い.

3.3.1 CUDA

CUDA とは NVIDIA が提供する GPGPU の開発環境である. NVIDIA の GPU を持っていないと使用することができないのでいわゆるベンダーロックインされる.

CUDA ででの基本的な処理の流れとしては

1. 使用するデバイス (GPU) の選択
2. デバイス上のメモリの確保 (cudaMalloc)
3. カーネルというデバイス上で実行させる処理を走らせる
4. デバイス上のメモリからホストのメモリに結果をコピーする (cudaMemcpy)



http://www.nvidia.co.jp/docs/IO/51174/NVIDIA_CUDA_Programming_Guide_1.1_JPN.pdf より引用

例: ベクトルの定数倍

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>

#define NUM 10

__global__ void prod(const int num, int *v)
{
    int i = threadIdx.x;
    v[i] = num * v[i];
}

int main()
{
    int v[NUM]{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    int *dev_v;
```

```
    cudaError_t cudaStatus;

    cudaStatus = cudaSetDevice(0);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaSetDevice failed");
        return -1;
    }

    cudaStatus = cudaMalloc((void**)&dev_v, NUM * sizeof(int));
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMalloc failed");
        return -1;
    }

    cudaStatus = cudaMemcpy(dev_v, v, NUM * sizeof(int),
        cudaMemcpyHostToDevice);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMemcpy failed");
        return -1;
    }

    prod << <1, NUM >> > (10, dev_v);

    cudaStatus = cudaMemcpy(v, dev_v, NUM * sizeof(int),
        cudaMemcpyDeviceToHost);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMemcpy failed");
        return -1;
    }

    // v {10, 20, 30, 40, 50, 60, 70, 80, 90, 100
}
```

レイトレーシングを cuda で行う場合は基本となるブロックとスレッドについて考えなればいけない．ブロックというのはスレッドの複数含んだものでスレッドというのは処理の最小単位である．課題となるのは 幅 × 高さ 回の計算をどのようにするかである．スレッド数には上限がありとても 1 ブロック n スレッドでは処理する量に限界がある．そこで N 回の処理には以下のようにしてカーネルを走らせる必要がある．

```
// デバイスで実行する処理
__global__ void kernel(Rgb *rgbs) {
    int index = threadIdx.x + blockIdx.x * blockDim.x;
    int w = index / (int)WIDTH_NUM;
    int h = index % HEIGHT_NUM;
    rgbs[index] = calc(w, h);
}

// 1ブロック128スレッドに固定する
kernel <<<(N + 127) / 128, 128 >>>(d_hoge);

// 1ピクセルあたりの処理は以下のようにする
kernel <<<(WIDTH_NUM*HEIGHT_NUM + 127) / 128, 128 >>>(d_rgbs);
```

上記のようにピクセルごとの処理を並列計算すればかなり早くレイトレーシングをレンダリングすることができる．とくにレイマーチングなどではピクセルあたりにかなり処理を回しているので単純に逐次

処理をしていては非常に処理が重くなってしまう.

3.4 constexpr

ここではコンパイル時にすでに計算してしまって実行時には出力するだけという方法を試してみる

コンパイル時計算は以下のような TMP など存在するがもっと簡単に実装できる

```
#include <iostream>

template<int N>
struct Factorial{
        static const int value = N * Factorial<N-1>::value;
};

template<>
struct Factorial<0>{
        static const int value = 1;
};

int main(){
        static_assert(Factorial<4>::value == 24, "error");
        std::cout << Factorial<4>::value << std::endl; // 24
        return 0;
}
```

3.4.1 Constant expressions(定数式)

定数, 関数, リテラルとして振る舞うクラスを定義することができる

```
constexpr int inc(int x){
    return x+1;
}
```

```
constexpr int num = 10;
int array[num];
```

constexpr は C++11 では関数が実質 return 文だけで定義する必要があり三項演算子, 再帰などを用いて関数を定義するしかなかったが C++14 では大幅な制限緩和がなされた

- 変数宣言を許可
- if 文と switch 文を許可
- 全てのループ文を許可 (for 文、範囲 for 文、while 文、do-while 文)
- 変数の書き換えを許可
- 戻り値型 (リテラル型) として、void を許可
- constexpr 非静的メンバ関数の、暗黙の const 修飾を削除

コンパイル時に動的に配列を生成する方法.

```
template <typename T, int LENGTH>
```

```
struct array{
    T v[LENGTH];

    constexpr array() : v(){
        for(int i=0;i<LENGTH;i++){
            this->v[i] = getSomething(i);
        }
    }

    constexpr T& operator[](size_t l){
        return v[l];
    }
    constexpr T const& operator[](size_t l)const{
        return v[l];
    }
};
```

上記の配列生成を多次元にして 1 ピクセルごとの計算を constexpr 対応の関数で計算すればコンパイル時レイトレーシングは可能となる.

コンパイル時にあらかしめ計算しておくことでほぼ IO などの処理しかかからないので爆速なのだが環境によるがコンパイル時間が n 時間かかったりバイナリのサイズが非常に大きくなってしまいます. そもそもコンパイル時にメモリが足らずコンパイルできなかったりします. 冷静に考えるとレイトレーシングをコンパイル時に計算することはあまり効果的とは言えないかもしれません.

参考文献

[1] http://www.nvidia.co.jp/docs/IO/51174/NVIDIA_CUDA_Programming_Guide_1.1_JPN.pdf

[2] https://www.ruche-home.net/program/bmp

[3] ゲームプログラミングのための 3D グラフィックス数学 Eric Lengyel

第 4 章

zsh_history 大公開！

Tachibana waita

概要
とりあえず部誌のにぎやかし要員を頼まれたので、にぎやかし用に自分の zsh_history を公開します。

4.1　概要

私はシェルとして zsh と呼ばれている物を使っています。MMA ではその昔 zsh 撲滅派によって 1 度削除された経歴がある歴史のあるシェルで、部員の中でも利用率が高いようです。

先ほど調べたところ、MMA コアサーバの中で bash 以外を利用しているユーザが 188 人おり、そのうち zsh を利用しているユーザは 34 人らしいです。デフォルトシェルをカスタマイズしている人は 35 人居て、例外の 1 人は fish を利用しているようでした。

4.2　zshrc の設定

さて、zsh における history 系の設定はそれなりにありますが、とりあえず以下のように設定すれば history ファイルは出力されます。各数値などは皆さんの環境に依存するのでよく確認しながら設定してください。

```
HISTFILE=~/.zsh_history
HISTSIZE=1000
SAVEHIST=100000000
setopt bang_hist
setopt extended_history
setopt hist_ignore_dups
setopt hist_reduce_blanks
```

この設定を入力する事で *~/.zsh_history* に history が出力されます。

4.3 history 公開

さて、本編として history を公開します。

```
: 1478703095:58;ssh home
: 1478703257:29;ssh home -Y
: 1478792156:61;ssh -vvv home
: 1478703343:16;ssh -Y root@192.168.53.5
: 1478792395:43;ssh home
: 1478824832:0;ls -l
: 1478824835:0;cd git
: 1478824835:0;ls -l
: 1478824841:0;cd misc-scripts
: 1478824842:0;ls -l
: 1478824844:0;git status
: 1478824848:26;git remote show origin
: 1478824882:5;less .git/config
: 1478824897:0;cd ../
: 1478824899:0;ls -l
: 1478824905:0;rm -rf Gifzo*
: 1478824906:0;ls -l
: 1478826475:0;ls -l
: 1478826477:0;ls-l
: 1478826478:0;ls -l
: 1478826480:30;open slide.pdf
: 1478832799:57;ssh home
: 1478833067:4818;ssh home
: 1478675656:0;docker ps
: 1478675660:0;docker ps -a
: 1478675664:0;docker images
: 1478675850:0;cd git
: 1478675851:0;ls -l
: 1478675854:0;mkdir docker-**
: 1478675855:0;cd docker
: 1478675864:0;cd docker-**
: 1478675865:0;ls -l
: 1478675878:1;cp ../docker-latex-jlisting/Dockerfile .
```

```
: 1478675879:0;ls -l
: 1478675880:99;vim Dockerfile
: 1478675988:40;docker build -t local/** .
: 1478676042:0;docker exec -it local/** /bin/bash
: 1478676047:103;docker run -it local/** /bin/bash
: 1478676155:7;vim Dockerfile
: 1478676163:15;docker build -t local/** .
: 1478676217:0;ls -l
: 1478676223:0;docker ps
: 1478676234:0;docker images
: 1478676247:163348;docker run -it **-standalone /bin/bash
: 1478839497:0;ls -l /private/tmp/com.apple.launchd.KpQzva2W9P/
: 1478839501:0;rm -rf /private/tmp/com.apple.launchd.KpQzva2W9P/
: 1478839563:18;sudo reboot
: 1478840133:30;ssh home
: 1478843450:7;tftp 192.168.53.16
: 1478843457:0;ls -l
: 1478843459:1;less test
: 1478843461:1;rm test
: 1478839996:0;ssh home
: 1478848975:30;tftp 192.168.53.16
: 1478849006:0;ls -l
: 1478849010:2;less undionly.kpxe
: 1478849014:0;rm -rf undionly.kp*
: 1478849016:1;rm ipxe
: 1478849019:0;ls -l
: 1478849421:0;curl http://192.168.53.16/boox.ipxe
: 1478849425:1;curl http://192.168.53.16/boot.ipxe
: 1478856203:13406;ssh -Y home
: 1478871260:2860;ssh home
: 1478878323:52;ssh -X home
: 1478886250:7;telnet 192.168.53.1
: 1478886267:83;ping 192.168.53.254
: 1478887396:0;ssh step
: 1478887398:1479;ssh home
: 1478922375:1695;ssh nest
: 1478924072:275;ssh home
```

```
: 1478924350:0;cd Documents
: 1478925642:0;ls -l
: 1478925808:3862;ssh nest
: 1479023443:0;ls -l
: 1479024105:2741;ssh home
: 1479027738:39;traceroute 8.8.8.8
: 1478847917:6902;ssh home
: 1478854910:0;date --date='+1 year' | grep -o '.惻惹'
: 1478854914:6;ssh sol
: 1478921152:47201;ssh home
: 1479023714:0;cd
: 1479024057:0;ls -l
: 1479024061:0;mkdir documents
: 1479024062:0;cd documents
: 1479024063:0;ls -l
: 1479024173:292;vim build.md
: 1479025611:0;dig chofutech.github.io
: 1479025985:0;dig chofu.tech
: 1479025994:0;dig chofu.tech a
: 1479026137:0;dig chofu.tech ns
: 1479026150:0;dig chofu.tech a
: 1479026156:0;dig chofu.tech
: 1479026169:0;dig chofu.tech ns
: 1479026255:0;dig chofu.tech a
: 1479026277:0;dig chofu.tech ns
: 1479026300:1;dig chofu.tech a
: 1479026306:0;dig chofu.tech
: 1479026348:1;dig chofu.tech a
: 1479026503:0;dig chofu.tech ns
: 1479026504:0;dig chofu.tech
: 1479026506:0;dig chofu.tech ns
: 1479026537:2;sudo dscacheutil -flushcache
: 1479026542:0;dig chofu.tech ns
: 1479026543:1;dig chofu.tech a
: 1479027041:2;whois chofu.tech
: 1479027083:0;ls -l
: 1479027095:2;ping 192.168.53.254
```

```
: 1479027100:11;telnet 192.168.53.254
: 1479027240:0;ls -l
: 1479027241:0;ll
: 1479027250:1332;ssh home
: 1479028265:318;ssh home
: 1479029610:0;cd tmp
: 1479029618:10346;vim electronic.md
: 1479039972:0;telnet 192.168.53.254
: 1479039987:0;telnet 192.168.53.253
: 1479042189:0;ls -l
: 1479041255:12;ping 192.168.53.253
: 1479041278:1;telnet 192.168.53.253
: 1479041302:39;ping 192.168.53.253
: 1479041375:1613;ssh home-pi
: 1479043976:0;cd git
: 1479043981:0;rm -rf docs/
: 1479044398:0;ping -f
: 1479044404:4;man ping
: 1479044424:4;sudo ping -f 10.1.2.3
: 1479044440:5;sudo ping -f 192.168.53.253
: 1479045099:47;ssh home
: 1479044240:7;telnet 192.168.53.254
: 1479044250:2814;telnet 192.168.53.253
: 1479047067:30;ssh home
: 1479043479:0;cd git
: 1479043522:0;ls -l
: 1479043524:0;cd images
: 1479043526:0;ls -l
: 1479043529:0;cd denari01
: 1479043529:0;ls -l
: 1479043530:0;open dentoo.lt.ai
: 1479043571:0;ls -l
: 1479043573:0;open dentoo.lt.
: 1479043574:0;open dentoo.lt.png
: 1479043957:3115;ssh home
: 1479047406:0;cd
: 1479047408:0;cd git/mi
```

```
: 1479047411:0;cd git/mikutter
: 1479047412:1;git pull
: 1479047434:0;diff mikutter.rb
: 1479098868:0;ls -l
: 1479098870:0;cd
: 1479098872:0;ssh home
: 1479099285:300;telnet 192.168.53.254
: 1479098897:5;ssh home-pi
: 1479103704:0;ls -l
: 1479103707:150;ssh home
: 1479103860:190;ssh -vvv home
: 1479104059:1068;ssh home
: 1479108869:1485;ssh home
: 1479106747:434;ssh home
: 1479107431:9;traceroute 8.8.8.8
: 1479107442:1323;ssh home
: 1479133749:33970;ssh home
: 1479204870:0;ls -l
: 1479204829:1;docker images
: 1479204843:1227;docker run -it **-standalone /bin/bash
: 1479129482:0;ls -l
: 1479168047:30;sudo vim /etc/hosts
: 1479168833:423;ssh home
: 1479193038:0;cd Server/
: 1479193039:0;ls -l
: 1479193045:0;mkdir **
: 1479193046:0;ls -l
: 1479193047:0;cd **
: 1479193047:0;ls -l
: 1479193050:0;cd ../
: 1479193052:0;rm -rf **
: 1479193053:0;cd
: 1479193054:0;cd git
: 1479193055:0;ls -l
: 1479193058:0;cd vagrant-**
: 1479193134:0;cd vagrant-**
: 1479193135:0;ls -l
```

```
: 1479193139:3;vim Vagrantfile
: 1479193145:188;vagrant up node2
: 1479193418:5;vim Vagrantfile
: 1479193426:722;vagrant up node2
: 1479196950:5;vagrant ssh
: 1479196961:24;vagrant ssh node2
: 1479196986:0;ls -l
: 1479196987:8;vim bootstrap-ubuntu.sh
: 1479196996:0;ls -l
: 1479197002:725;vagrant ssh node2
: 1479198778:0;ls -l
: 1479199439:3;view bootstrap-ubuntu.sh
: 1479199444:1266;vim bootstrap-ubuntu.sh
: 1479200713:0;ls -l
: 1479200714:0;cd **-plugin-**
: 1479200715:0;ls -l
: 1479200718:9;less CMakeLists.txt
: 1479201052:0;cd ../
: 1479201053:0;ls -l
: 1479201056:1;less bootstrap-ubuntu.sh
: 1479201061:167;vim bootstrap-ubuntu.sh
: 1479201229:33;vim Vagrantfile
: 1479201265:5;vagrant reload node2
: 1479201273:8;vim Vagrantfile
: 1479201282:31;vagrant reload node2
: 1479201343:338;vagrant provision node2
: 1479204400:0;ls -l
: 1479204403:4;vim bootstrap-ubuntu.sh
: 1479204408:3;git diff
: 1479204412:0;git status
: 1479204417:6;git diff bootstrap-ubuntu.sh
: 1479204425:0;git add bootstrap-ubuntu.sh
: 1479204427:5;git commit
: 1479204439:0;git diff Vagrantfile
: 1479204497:23;git add -p Vagrantfile
: 1479204525:0;git status
: 1479204529:0;git diff Vagrantfile
```

```
: 1479204534:8;git commit
: 1479204543:1;git push
: 1479206338:23;vim bootstrap-ubuntu.sh
: 1479206382:7641;less bootstrap-ubuntu.sh
: 1479215125:0;ls -l
: 1479215127:2;vim bootstrap
: 1479215130:8;vim bootstrap-ubuntu.sh
: 1479215147:93;vagrant provision node2
: 1479215398:1478;less bootstrap-ubuntu.sh
: 1479216879:3;vim bootstrap-ubuntu.sh
: 1479216897:259;vagrant provision node3
: 1479217236:6;vagrant ssh
: 1479217245:57;vagrant ssh node3
: 1479217305:15;vim bootstrap-ubuntu.sh
: 1479217322:23;vagrant ssh node3
: 1479217346:0;ls -l
: 1479217349:21;vim bootstrap-ubuntu.sh
: 1479217379:400;vagrant destroy -f node3 ; vagrant up node3
: 1479217942:4;vim bootstrap-ubuntu.sh
: 1479218128:60;vagrant provision node3
: 1479218277:0;ls -l
: 1479218279:21;less bootstrap-ubuntu.sh
: 1479218320:0;cat bootstrap-ubuntu.sh
: 1479273102:1;whois painfuldance.asia
: 1479273624:1;ls -l
: 1479273629:0;cd git/vagrant-**
: 1479273629:0;ls -l
: 1479197203:0;cd git/vagrant-**
: 1479197203:0;ls -l
: 1479197205:0;-vim bootstrap
: 1479197208:1;vim bootstrap
: 1479197211:247;vim bootstrap-ubuntu.sh
: 1479197523:2;vim bootstrap
: 1479197525:180;vim bootstrap-ubuntu.sh
: 1479197708:0;ls -l
: 1479197711:1;rm -rf **
: 1479197721:988;vagrant destroy -f node2 ; vagrant up node2
```

```
: 1479198762:5;vagrant ssh
: 1479201350:14749;vagrant ssh node2
: 1479216100:0;ls -l
: 1479216104:4;vim Vagrantfile
: 1479216109:3;vagrant ssh
: 1479216114:19;vagrant ssh node2
: 1479216134:0;ls -l
: 1479216136:27;vim Vagrantfile
: 1479216827:9;vagrant up node3
: 1479216840:47;vagrant ssh node3
: 1479217293:6;vim bootstrap-ubuntu.sh
: 1479217912:161;vagrant ssh node3
: 1479218079:47;vim bootstrap-ubuntu.sh
: 1479273237:178;vagrant ssh node3
: 1479273419:13;vagrant halt node2
: 1479273634:0;ls -l
: 1479273651:26;vim **-plugin-interface.h
: 1479273742:1;vim **-plugin-
: 1479273745:3;vim **-plugin-interface.h
: 1479273751:0;cat **-plugin-interface.h
: 1479274395:0;cd git/vagrant-**
: 1479274395:0;ls -l
: 1479274398:0;git status
: 1479274403:0;git add **-plugin-interface.h
: 1479274408:6;git commit
: 1479274426:0;ls -l
: 1479274435:0;cd git/vagrant-**
: 1479274436:0;ls -l
: 1479274441:11;view bootstrap-ubuntu.sh
: 1479274454:12;vim bootstrap-ubuntu.sh
: 1479273692:0;cd git/vagrant-**
: 1479273695:2882;vagrant ssh node3
: 1479281325:36;vagrant halt node3
: 1479282668:29;vagrant up node3
: 1479282738:1187;vagrant ssh node3
: 1479283926:0;ls -l
: 1479283929:2;vim ~/bin
```

```
: 1479283933:0;cd ~/bin
: 1479283934:0;ls -l
: 1479283941:1;wget https://raw.githubusercontent.com/masuidrive/vagrant-remote/master/vagr
: 1479283942:0;ls -l
: 1479283946:0;chmod +x vagrant-remote
: 1479283963:0;ls -l
: 1479283968:0;chmox -x vagrant-remote
: 1479283970:0;chmod -x vagrant-remote
: 1479283971:0;ls -l
: 1479283978:0;cd ../
: 1479283990:33;vim ~/.local/zsh.local
: 1479284026:0;. ~/.zshrc
: 1479284052:11;vim ~/.local/zsh.local
: 1479284069:0;ls -l /Users/whywaita/bin/vagrant-remote
: 1479284077:12;vim ~/.local/zsh.local
: 1479284094:0;. ~/.zshrc
: 1479284097:5;vim ~/.local/zsh.local
: 1479284103:0;. ~/.zshrc
: 1479284160:0;ls -l
: 1479284161:0;cd bin
: 1479284162:0;ls -l
: 1479284166:0;chmod +x vagrant-remote
: 1479284167:0;cd
: 1479284180:0;cd git/vagrant-**
: 1479284180:0;ls -l
: 1479284185:21;vim .vagrant-remote
: 1479284226:3;vim .vagrant-remote up node3
: 1479284233:0;vagrant-remote up node3
: 1479286371:59;shell-alias
: 1479284245:0;cd git/vagrant-**
: 1479284247:0;vagrant-remote up node3
: 1479284255:0;ls -l ~/bin/
: 1479284262:0;echo $PATH
: 1479284295:10;vim ~/.local/zsh.local
: 1479284315:0;mv ~/bin/vagrant-remote ~/opt/
: 1479284322:8;vagrant-remote up node3
: 1479284351:9;vagrant halt node3
```

```
: 1479284364:1;vagrant-remote up node3
: 1479284391:0;ls -l /tmp
: 1479284393:0;ls -l /tmp/
: 1479284402:2;vim .vagrant
: 1479284405:27;vim .vagrant-remote
: 1479284434:2;vagrant-remote up node3
: 1479284441:6;vim .vagrant-remote
: 1479284448:0;ls -l /Users/whywaita/tmp/
: 1479284453:2;vagrant-remote up node3
: 1479284467:6;vim .vagrant-remote
: 1479284474:2;vagrant-remote up node3
: 1479284534:19;vim Vagrantfile
: 1479284554:2;vagrant-remote up node3
: 1479284559:4;vim Vagrantfile
: 1479284566:3;vim .vagrant-remote
: 1479284591:11;vim Vagrantfile
: 1479284608:46;vim ~/opt/vagrant-remote
: 1479284667:1;vim ~/opt/vagrant-remote
: 1479284670:7;vim .vagrant-remote
: 1479284681:3;vagrant-remote up node3
: 1479284705:22;vim ~/opt/vagrant-remote
: 1479284731:3;less /etc/exports
: 1479284736:0;sudo nfsd update
: 1479284738:9;vim ~/opt/vagrant-remote
: 1479284754:2;vagrant-remote up node3
: 1479284765:26;vim ~/opt/vagrant-remote
: 1479284792:0;ifconfig|grep "inet $NETWORK"|sed -E "s/.*inet[\t ]*([0-9.]+).*/\1/g"|head -
: 1479284798:0;ifconfig|grep "inet $NETWORK"|sed -E "s/.*inet[\t ]*([0-9.]+).*/\1/g"
: 1479284804:0;ifconfig|grep "inet $NETWORK"|sed -E "s/.*inet[\t ]*([0-9.]+).*/\1/g"| tail
: 1479284807:7;vim ~/opt/vagrant-remote
: 1479284817:2;vagrant-remote up node3
: 1479284829:23;vim ~/opt/vagrant-remote
: 1479284884:2;vagrant-remote up node3
: 1479284915:0;ifconfig
: 1479284937:8;vim ~/opt/vagrant-remote
: 1479284947:210;vagrant-remote up node3
: 1479285207:25;vagrant up node3
```

```
: 1479285279:1;vagrant ssh node2
: 1479288628:14;vagrant ssh node3
: 1479288647:1;vagrant ssh-config node2
: 1479288649:4;vagrant ssh-config node3
: 1479288681:6;ssh root@localhost -p 2222 -i ~/.vagrant.d/insecure_private_key
: 1479288718:36;vagrant ssh node3
: 1479288784:1;scp -p 2222 -i ~/.vagrant.d/insecure_private_key vagrant@localhost:**-**.con
: 1479288798:0;scp vagrant@localhost:**-**.config.xml . -p 2222
: 1479288805:0;scp -P 2222 -i ~/.vagrant.d/insecure_private_key vagrant@localhost:**-**.con
: 1479288813:0;ls -l
: 1479288821:0;cp **-**.config.xml{,.org}
: 1479288825:21;vim **-**.config.xml
: 1479288881:0;ls -l
: 1479288887:1;vim **-**.config.xml
: 1479288946:0;ls -l
: 1479288959:0;mv **-**.config.xml **-**.config.graphml
: 1479288959:1;ls -l
: 1479289098:1622;vim **-**.config.graphml
: 1479288419:7;brew search gephi
: 1479288431:17;brew install gephi
: 1479288463:45;brew cask install gephi
: 1479291567:0;alias shell-alias
: 1479295993:0;cd tmp
: 1479295994:0;ls -l
: 1479297635:0;cd
: 1479297700:0;cd tmp
: 1479297716:8;vim **-denki.md
: 1479297732:0;mv **-denki.md ^^e5^^83^^a6訳.txt
: 1479297735:30;open ^^e5^^83^^a6訳.txt
: 1479297785:11;open .
: 1479300886:8;vim ~/.ssh/config
: 1479302796:7;vim ~/.ssh/config
: 1479303447:0;ls -l
: 1479303855:5;vim ~/.ssh/config
: 1479288916:0;cd git/vagrant-**
: 1479288919:2;vagrant ssh node2
: 1479288921:16937;vagrant ssh node3
```

```
: 1479307072:1026;ssh home
: 1479308116:0;. ~/.zshrc
: 1479041359:0;tns ba
: 1479041360:242668;ta ba
: 1479284031:1;tns ba
: 1479284045:3;vim ~/.local/zsh.local
: 1479284049:55;tns ba
: 1479284112:1;vim ~/.local/zsh.local
: 1479284116:29;vim bin/vagrant-remote
: 1479284147:5;vim ~/.local/zsh.local
: 1479308107:13;tns ba
: 1479308125:2;vim ~/.local/zsh.local
: 1479308319:0;ls -l
: 1479308325:4;ssh home
: 1479308330:0;ls -l
: 1479308417:6;vagrant global-status
: 1479308430:0;cd Server/**
: 1479308433:0;vagrantstatus
: 1479308435:7;vagrant status
: 1479308448:10;vagrant destroy -f
: 1479308462:5;vagrant global-status
: 1479308470:0;cd
: 1479308478:0;cd git/vagrant-gitlab
: 1479308481:7;vagrant status
: 1479308495:6;vagrant destroy -f
: 1479308759:0;cd
: 1479308760:0;ls -l
: 1479308761:0;cd git/vagrant-
: 1479308764:0;cd git/vagrant-**
: 1479308835:0;ls -l
: 1479308842:0;git status
: 1479308851:0;rm -rf .vagrant-remote
: 1479308855:0;rm -rf ~/opt/vagrant-remote
: 1479308856:0;ls -l
: 1479308858:0;git status
: 1479308861:2;less insecure_private_key
: 1479308870:0;diff insecure_private_key ~/.vagrant.d/insecure_private_key
```

```
: 1479308871:1;rm insecure_private_key
: 1479308873:0;ls -l
: 1479308876:0;git status
: 1479308880:0;git diff bootstrap-ubuntu.sh
: 1479361540:0;cd git/vagrant-**
: 1479361541:0;ls -l
: 1479361542:12;less bootstrap-ubuntu.sh
: 1479361557:0;cat bootstrap-ubuntu.sh
: 1479362940:35;vagrant up node2
: 1479362985:76;vagrant ssh node2
: 1479361690:0;cd git
: 1479361692:1;git clone https://github.com/whywaita/**
: 1479361697:18;git clone https://github.com/whywaita/**
: 1479361733:0;ls -l
: 1479361734:0;cd **
: 1479361735:0;ls -l
: 1479361769:0;cd ../
: 1479361789:0;cd **
: 1479361790:0;ls -l
: 1479361798:11;vim main.cpp
: 1479361810:0;cd ../
: 1479361812:0;rm -rf **
: 1479380275:0;autoload -U tcp_open
: 1479380293:0;tcp_open post-3.cc.uec.ac.jp 80 ; echo $?
: 1479380297:0;tcp_open post-3.cc.uec.ac.jp 443 ; echo $?
: 1479380300:76;tcp_open post-3.cc.uec.ac.jp 567 ; echo $?
: 1479308885:0;tns ba
: 1479402811:4;ping 192.168.2.1
: 1479407984:23364;ssh home
: 1479437193:0;ls -l
: 1479438412:0;cd git/vagrant-
: 1479438414:0;cd git/vagrant-**/
: 1479438414:0;ls -l
: 1479438934:30;shell-alias
: 1479444365:0;cd git/vagrant-**
: 1479444365:0;ls -l
: 1479444370:14;vim diff.patch
```

```
: 1479444385:0;ls -l
: 1479452797:0;cd git/vagrant-**
: 1479452798:0;git status
: 1479452801:0;git diff Vagrantfile
: 1479452814:57;git add -p Vagrantfile
: 1479452873:0;git status
: 1479452875:0;git diff Vagrantfile
: 1479452877:0;ls -l
: 1479452880:6;git commit
: 1479452887:0;git psuh
: 1479452890:1;git push
: 1479456513:188;ssh home
: 1479464703:0;cd git/vagrant-**
: 1479464703:0;ls -l
: 1479464707:43;vagrant ssh node3
: 1479466251:49;ssh sol
: 1479465974:331;ssh home
: 1479485132:204;shell-alias
: 1479485256:4;telnet 192.168.53.254
: 1479485261:106;telnet 192.168.53.253
: 1479464760:0;cd git/vagrant-**
: 1479464761:0;ls -l
: 1479464769:19739;vagrant ssh node3
: 1479536718:127;telnet 192.168.53.253
: 1479543661:337;ssh sol
: 1479485058:0;tns ba
: 1479485059:0;ta ba
: 1479391771:0;cd git
: 1479391771:0;ls -l
: 1479391843:0;ls -l | grep 2014
: 1479392348:0;ls -l
: 1479392350:0;cd itc2014
: 1479392350:0;ls -l
: 1479392357:0;mkdir submit-documents
: 1479392358:0;cd submit-documents
: 1479392359:0;ls -l
: 1479392375:0;git config user.email
```

```
: 1479392379:0;git config user.name
: 1479392389:0;ls -l
: 1479392406:297;vim submit-document.tex
: 1479392709:1;platex submit-document.tex
: 1479392715:0;ls -l
: 1479392731:1;dvipdfmx submit-document.dvi
: 1479392734:0;open submit-document.
: 1479392735:0;open submit-document.pdf
: 1479392764:35;vim submit-document.tex
: 1479392814:0;platex submit-document.tex; dvipdfmx submit-document.dvi
: 1479392820:1;platex submit-document.tex; dvipdfmx submit-document.dvi ; open submit-docum
: 1479392830:17;vim submit-document.tex
: 1479392849:0;platex submit-document.tex; dvipdfmx submit-document.dvi ; open submit-docum
: 1479392859:264;vim submit-document.tex
: 1479393125:0;git status
: 1479393128:0;ls -l
: 1479393136:0;git add submit-document.tex submit-document.pdf
: 1479393140:10;git commit -S
: 1479393156:1;git push -u origin master
: 1479393197:0;cd ../
: 1479393198:0;ls -l
: 1479393207:149;vim .gitlab-ci.yml
: 1479393359:0;git add .gitlab-ci.yml
: 1479393360:7;git commit -S
: 1479393367:2;git push
: 1479393371:0;cd submit-documents
: 1479393373:0;ls -l
: 1479393376:88;vim submit-document.tex
: 1479393467:0;git status
: 1479393471:1;platex submit-document.tex; dvipdfmx submit-document.dvi ; open submit-docum
: 1479393509:0;cd ../
: 1479393510:0;ls -l
: 1479393512:0;cat .git
: 1479393513:0;cat .gitlab-ci.yml
: 1479393517:6;vim .gitlab-ci.yml
: 1479393524:0;cat .gitlab-ci.yml
: 1479393541:0;git status
```

```
: 1479393543:0;cd ../
: 1479393549:0;cd itc2014
: 1479393550:0;ls -l
: 1479393553:0;git add submit-documents/submit-document.tex
: 1479393555:8;git commit
: 1479393566:0;git add .gitlab-ci.yml
: 1479393569:6;git commit -S
: 1479393579:13;vim .gitignore
: 1479393595:0;git add .gitignore
: 1479393596:4;git commit -S
: 1479393602:1;git push
: 1479393951:0;git status
: 1479393962:0;git rm --cached submit-documents/submit-document.pdf
: 1479393970:0;ls -l
: 1479399728:1;git pull
: 1479399738:0;ls -l
: 1479399755:0;open materials/explanatory-material.pptx
: 1479402601:2132;ssh nest
: 1479404735:0;ls -l
: 1479404736:0;git pull
: 1479404742:0;open submit-documents/submit-document.pdf
: 1479404824:0;cd submit-documents
: 1479404824:0;ls -l
: 1479404827:5;vim submit-document.tex
: 1479404835:0;git add submit-document.tex
: 1479404837:20;git commit -S
: 1479404859:0;git push
: 1479405000:0;ls -l
: 1479405002:0;cd ../
: 1479407802:23546;ssh home
: 1479431440:0;open .
: 1479431446:6;git pull
: 1479432177:1210;ssh nest
: 1479436014:0;ls -l
: 1479436017:0;cd ../
: 1479436127:0;cd vagrant-**
: 1479436127:0;ls -l
```

```
: 1479436130:7;vagrant ssh node3
: 1479436143:25;vagrant up node3
: 1479436179:484;vagrant ssh node3
: 1479436666:34;vagrant up node2
: 1479436721:309;vagrant ssh node2
: 1479437032:7;vagrant ssh node3
: 1479437041:248;vagrant ssh node2
: 1479437290:0;ls -l
: 1479437293:16;vagrant ssh node2
: 1479437311:1791;vagrant ssh node3
: 1479439128:11;vagrant halt node2
: 1479439184:0;ls -l
: 1479443613:8896;vagrant ssh node3
: 1479452512:0;ls -l
: 1479452514:99;vim bootstrap-ubuntu.sh
: 1479452614:1;git status
: 1479452622:0;git add diff.patch
: 1479452624:15;git commit -S
: 1479452640:0;ls -l
: 1479452643:2;vim bootstrap-ubuntu.sh
: 1479452647:0;git diff bootstrap-ubuntu.sh
: 1479452654:0;git add bootstrap-ubuntu.sh
: 1479452655:14;git commit -S
: 1479452673:0;git pull
: 1479452675:2;git push
: 1479453367:0;;ls -l
: 1479453381:5;vagrant status
: 1479453393:3;vagrant node3
: 1479453480:31058;vagrant ssh node3
: 1479534462:0;vagrant ssh node3
: 1479545325:150;ssh sol
: 1479545476:486;ssh home
: 1479546053:1;ping 192.168.12.254
: 1479546062:220;telnet 192.168.12.254
: 1479546287:1;ping 192.168.12.51
: 1479546295:10;ssh sol
: 1479546882:511;vim /Users/whywaita/synergy.conf
```

```
: 1479547579:194;ssh sol
: 1479547832:468;ssh sol
: 1479548638:2;ping 172.21.66.172
: 1479548643:1;ssh 172.21.66.172
: 1479548666:0;ssh 192.168.12.254
: 1479548682:3;telnet 192.168.12.254
: 1479548689:1;ping 172.21.66.172
: 1479548694:2;ping 192.168.12.254
: 1479548715:2405;ssh sol
: 1479547592:4857;ssh home
: 1479553437:0;alias nnmap
: 1479555474:51;ssh nest
: 1479555530:0;ls -l
: 1479555534:16;vim ~/.ssh/config
: 1479555569:0;ls -l
: 1479555596:55;vim ~/.ssh/config
: 1479555661:12;vim ~/.ssh/config
: 1479555681:21;vim ~/.ssh/config
: 1479553396:12;ssh sol
: 1479553411:7;ssh moon
: 1479553420:2312;ssh nest
: 1479545210:111;ssh home
: 1479545322:0;tns ba
: 1479547654:290;telnet 192.168.12.254
: 1479547945:4;ssh 192.168.12.51
: 1479547951:4503;telnet 192.168.12.254
: 1479552740:0;telnet 192.168.12.254
: 1479552989:47;ssh home
: 1479553037:0;ls -l
: 1479553058:6136;ssh home
: 1479563349:5;vim ~/.ssh/config
: 1479563355:0;cd .ssh
: 1479563357:0;git status
: 1479563360:2;git diff config
: 1479563365:0;git add config
: 1479563368:11;git commit -S
: 1479563381:0;git status
```

```
: 1479564578:0;cd ../
: 1479579309:0;ls -l
: 1479579312:0;git status
: 1479579314:0;cd .ssh
: 1479579316:0;git status
: 1479579319:17;git diff known_hosts
: 1479579339:31;ssh ajou
: 1479579381:216;ssh home
: 1479580197:1;ssh nest
: 1479580208:76;ssh lab-arch
: 1479553880:0;cd git/vagrant-**
: 1479553885:6;vagrant ssh node3
: 1479553915:26;vagrant up node3
: 1479553946:63103;vagrant ssh node3
: 1479639480:1;ls -l
: 1479639489:0;pwd
: 1479639560:219;ssh home
: 1479632883:6468;ssh home
: 1479639542:0;mkdir -p /Volumes/Share/backup/vmware-fusion
: 1479642288:2;ping google.com
: 1479642294:2;ping 192.168.53.1
: 1479642613:0;scp mac
: 1479642619:0;cd Documents/
: 1479642689:0;cd 仮惣ャシャ
: 1479642779:0;cd Virtual\ Machines.localized
: 1479642779:0;ls -l
: 1479642812:6;ssh 192.168.53.100
: 1479642842:3081;scp -r Windows\ 7\ x64.vmwarevm 192.168.53.100:
: 1479649525:0;cd
: 1479649531:3486;ssh home
: 1479652887:0;open Downloads
: 1479653158:33554;ssh nest
: 1479693491:17;ssh home
: 1479693526:0;cd git/vagrant-**
: 1479693526:0;ls -l
: 1479693528:5;vagrant ssh
: 1479693537:4;vagrant ssh node3
```

```
: 1479693559:26;vagrant up node3
: 1479693588:793;vagrant ssh node3
: 1479694382:0;git status
: 1479695417:0;vagrant --version
: 1479695579:0;cd ../lab
: 1479695581:1;git pull
: 1479695592:0;cd dayly-document
: 1479695593:0;ls -l
: 1479695594:265;vim 20161121.md
: 1479695865:0;pandoc 20161121.md 20161121.html
: 1479695892:1;pandoc 20161121.md -o 20161121.html
: 1479695894:0;ls -l
: 1479695896:0;open 20161121.html
: 1479695905:19;vim 20161121.html
: 1479695925:0;open 20161121.html
: 1479695934:0;git add 20161121.*
: 1479695935:11;git commit
: 1479695948:1;git push
: 1479713645:174;ssh home
: 1479716806:18;shell-alias
: 1479714235:2168;shell-alias
: 1479716406:423;ssh nest
: 1479718603:3;ssh home
: 1479718608:459;ssh nest
: 1479785111:0;cd git/vagrant-**
: 1479785117:12;vagrant halt node3
: 1479785505:0;cd git/vagrant-**
: 1479785507:0;ls -l
: 1479785516:18;vim bootstrap-ubuntu.sh
: 1479786090:0;l -l
: 1479786090:0;ls -l
: 1479786095:0;cd git/vagrant-**
: 1479786096:0;ls -l
: 1479786099:0;cat diff.patch
: 1479786131:0;ls -l
: 1479786132:0;cat shd-library.h
: 1479786139:0;cat **-plugin-interface.h
```

```
: 1479786171:0;cat shd-library.h
: 1479806761:0;cd git/vagrant-sa
: 1479806762:0;cd git/vagrant-**
: 1479806762:0;ls -l
: 1479806763:1091;less bootstrap-ubuntu.sh
: 1479812326:189;ssh nest
: 1479812622:0;ls -l
: 1479802041:0;ls -l
: 1479802047:0;cd git/vagrant-
: 1479802049:0;cd git/vagrant-sha
: 1479802050:0;cd git/vagrant-**
: 1479802051:0;ls -l
: 1479802054:19590;vagrant ssh node4
: 1479825473:1;whois 211.10.27.230
: 1479825488:2;whois 211.10.27.230 | nkf -wJ
: 1479825495:1;whois 61.213.140.162
: 1479846603:7;vagrant status
: 1479846618:10;vagrant halt node4
: 1479846700:12;vagrant halt node3
: 1479846738:8;vagrant status
: 1479899305:0;history
: 1479899314:13;less ~/.zsh_history
: 1479899328:0;ls -l
: 1479899329:2;less ~/.zsh_history
: 1480132115:1;scp Desktop/kaki.pdf nest:
: 1480132118:0;ls -l
: 1480132119:551;ssh nest
: 1480210598:0;cd git/booklet2016s
: 1480210599:0;ls -l
: 1480210608:0;cd ../
: 1480210613:0;rm -rf booklet2016s
: 1480210614:0;ls -l
: 1480211174:377;ssh nest
: 1479965975:0;cd dotfiles
: 1479965976:1;git status
: 1479965983:0;git status .gitconfig
: 1479965987:84;vim .zshrc
```

```
: 1479966074:0;git add .zshrc r
: 1479966075:0;git add .zshrc
: 1479966085:15;git commit -S
: 1479966102:5;git push
: 1479966112:0;history -s
: 1479966113:0;history -d
: 1479966143:0;ls -l
: 1479966145:0;git status
: 1479966151:0;git diff .gitconfig
: 1479966157:0;cat index.html
: 1479966164:1;rm index.html
: 1479966166:0;ls -l
: 1479966167:0;git status
: 1479966177:0;git diff .vimrc.programing
: 1479966187:0;git add .vimrc.programing
: 1479966192:9;git commit -S
: 1479966203:1;git status
: 1479966216:3;vim .gitignore
: 1479966223:0;git add .gitignore
: 1479966225:16;git commit -S
: 1479966243:4;git push
: 1479966249:0;git status
: 1479967661:0;cd
: 1479994624:12999;ssh home
: 1480044392:0;history -d
: 1480129587:0;ls -l
: 1480130455:0;cd git/
: 1480130456:0;ls -l
: 1480130469:0;cd
: 1480130473:0;cd Documents/slide/
: 1480130473:0;ls -l
: 1480130478:0;mkdir DentooLT15.5
: 1480130479:0;cd DentooLT
: 1480130482:0;cd DentooLT15.5
: 1480130482:0;ls -l
: 1480130485:0;touch slide.md
: 1480130486:0;ls -l
```

```
: 1480130492:0;mv slide.md otaku.md
: 1480130492:0;ls -l
: 1480130496:0;open otaku.md
: 1480130706:10;ssh tamaki
: 1480131415:1256;ssh nest
: 1480132678:0;mv ~/Desktop/stage.png .
: 1480136027:0;mv ~/Desktop/birth.png .
: 1480136031:0;mv ~/Desktop/yoko* .
: 1480136032:0;ls -l
: 1480137829:0;mv ~/Desktop/zinmyaku.png .
: 1480139381:0;open otaku.pdf
: 1480140422:0;open ~/Downloads/dentooLT16.pdf
: 1480141256:1;ssh 202.122.60.206
: 1480141259:46;ssh 202.122.60.206 -l pi
: 1480141467:4;ping 202.122.60.206
: 1480143727:0;open otaku.pdf
: 1480148178:1;open otaku.md
: 1480148440:0;open otaku.pdf
: 1480210500:0;cd
: 1480210501:0;cd git
: 1480210506:0;cd booklet2016c
: 1480210507:0;ls -l
: 1480210510:4;vim all.tex
: 1480210517:19;vim okuduke/okuduke.tex
: 1480210537:0;ls -l
: 1480210539:16;vim all.tex
: 1480210558:0;ls -l mizunashi
: 1480210570:0;cp -ra sample why
: 1480210574:0;cp -r sample why
: 1480210575:0;ls -l
: 1480210576:0;cd why
: 1480210578:0;ls -l
: 1480210579:215;vim sample.tex
: 1480210795:0;ls -l
: 1480210798:0;cd ../
: 1480210798:0;ls -l
: 1480210801:11;rake
```

```
: 1480210822:1;vim all.t
: 1480210824:7;vim all.tex
: 1480210833:2;rake
: 1480210854:9;brew search pygmentize
: 1480210877:19;brew install pygments
: 1480210926:1;rake
: 1480210934:0;open all.pdf
: 1480210936:0;ls -l
: 1480210958:0;git add okuduke/okuduke.tex all.tex
: 1480210961:9;git commit -S
: 1480210975:0;cd why
: 1480210977:0;ls -l
: 1480210985:0;mv sample.tex why.tex
: 1480210987:0;cd ../
: 1480210988:0;rake
: 1480210992:1;rake clean
: 1480210994:2;rake
: 1480210999:0;git status
: 1480211002:0;git add why/why.tex
: 1480211004:6;git commit -S
: 1480211012:1;git push
: 1480211048:0;ls -l
: 1480211052:688;vim why/why.tex
```

4.4　おわりに

これでひとまずおしまいです。マズい情報が入っていたら気にせず報告してください。

百萬石 2016 -秋- ©電気通信大学 MMA

2016 年 11 月 27 日　初版第一刷発行　【本書の無断転載を禁ず】

著　者　f_dita
mizunashi_mana
su_zu
Tachibana waita
表　紙　g_nootoko
編集者　hyr3k
発行者　電気通信大学 MMA
発行所　電気通信大学 MMA 部室
〒182-8585 東京都調布市調布ヶ丘 1-5-1 サークル会館 208 号室
https://www.mma.club.uec.ac.jp/
印刷所　電気通信大学 MMA 部室
製本所　電気通信大学 MMA 部室

百夢石 Hyakumangoku Vol.47

電気通信大学