2021
春号

百萬石
Hyakumangoku Vol.56

Published by

部長挨拶

terry

terry@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．

さて, この「春号」は新歓期間中に配布されるので, きっと多くの新入生の方に御覧頂いていることかと思います．そのため，ここからは少しだけサークルの特徴をご紹介したいと思います．

MMA は技術に強い人材がたくさんいます．しかし，**初心者お断りサークルでは決してありません**．そういった先輩や仲間に支えてもらいながら，自分の好きなこと，やってみたいことを実現していけるかもしれない魔法のようなサークルです．計算機 (コンピュータ) を使っていさえすれば，活動内容は自由です．ただ，このメリットは「やることを探さなくてはならない」というデメリットと表裏一体です．入ったはいいものの，なにをすればいいのかわからないがために，幽霊部員となっていく人もいます．

そこで我々部員は「新入生講習会」を用意しました．ここで，先輩との接点を持ち，雑談のような会話をしていく中で，自分の興味はどこにあるのかを探してもらい，やりたいことを見つけてもらおうと思っています．

これまでは部活の抽象的な特徴についてお話しましたが，具体的な活動内容を紹介します．部員全員が部室に出入り自由になるような電子錠システムの開発 (ソフトウェア開発)，ある制約の中での課題について，答えを導くプログラムを正確に作る速さを競う競技プログラミングの大会に参加することや，部内勉強会の開催などを行っています．

最後に，MMA はいろんな人とのつながり，自分のやってみたいことを相談できる環境，充実した機材が揃っていると，自分は感じます．そんな自由度の高いところで活動している部員が書く部誌は面白いものばかりだと思います．ぜひご一読ください．

2021 年 3 月吉日

百萬石 Vol.56

Hyakumangoku Vol.56

2021 Spring

MMA

目次

第 1 章

ソートの改良

tonphy

1.1 始めに

tonphy です。この記事はクイックソートとマージソートの改良について書いています。ざっくりとしたソートの説明はしますが詳しい説明はここではしないので、気になった方は自分で調べてみてください。

クイックソートは単純な改良ですが、マージソートはエレガントなコードになったので是非見て頂けると嬉しいです。

1.2 クイックソート

1.2.1 ナイーブな実装

クイックソートはピボットと呼ばれる暫定的な中央値を決めて、データをピボットよりも小さいデータと大きいデータに二分する、分割統治的なアルゴリズムです。この操作を分割されたデータに対して繰り返す事で、全体をソートする事ができます。

```
#include<iostream>
#define SWAP(a,b) {int temp; temp = a; a = b; b = temp;}
using namespace std;

void QuickSort(int* begin, int* end);

//半開区間
void QuickSort(int* begin, int* end){
  if(end - begin <= 1){
    return;
  }

  int* left = begin;
  int* right = end - 1;
  int pivot = *left;

  while(1){
    while(left < end && *left < pivot){
      left++;
```

```
    }
    while(begin <= right && *right > pivot){
      right--;
    }
    if(right < left) break;
    SWAP(*left, *right);
    left++;right--;
  }

  QuickSort(begin, right + 1);
  QuickSort(left, end);
  return;
}
```

1.2.2　ピボットの改良

このままでもクイックソートは十分速いのですが、ピボットをデータの中央値に近づける事でクイックソートの性能を上げる事ができます。複数の値 (n より小さくないと意味が無い) の平均値や中央値をピボットとする方法があります。

自分は右端、左端、中央のデータの平均値をピボットとするのがお気に入りです。これだけでもまぁまぁ速くなります。

```
#include<iostream>
#define SWAP(a,b) {int temp; temp = a; a = b; b = temp;}
using namespace std;

void QuickSort(int* begin, int* end);

//半開区間
void QuickSort(int* begin, int* end){
  if(end - begin <= 1){
    return;
  }

  int* left = begin;
  int* right = end - 1;
  //ここを変更するだけ
  int pivot = (*left + *right + *(begin + (end - begin) / 2) ) / 3;

  while(1){
    while(left <end && *left < pivot){
      left++;
    }
    while(begin <= right && *right > pivot){
      right--;
    }
    if(right < left) break;
    SWAP(*left, *right);
    left++;right--;
  }

  QuickSort(begin, right + 1);
  QuickSort(left, end);
  return;
}
```

1.3　マージソート

1.3.1　ナイーブな実装

マージソートは前半分と後ろ半分をそれぞれソートした後に、適切に元の配列にデータを詰め直す（マージ）事でソートを行います。ソートの特性上、データ数と同サイズの作業用配列が必要になります。

```
#include<iostream>
using namespace std;

void MergeSort(int* begin, int* end);
void mergesort(int* array, int* work, int n);

//半開区間
//配列確保の為に呼び出し
void MergeSort(int* begin, int* end){
  int n = end - begin;
  int *work = new int[n];
  mergesort(begin, work, n);
  return;
}

void mergesort(int* array, int* work, int n){
  if(n <= 1){
    return;
  }

  mergesort(array, work, n/2);//前半分をソート
  mergesort(array + n/2, work + n/2, n - n/2);//後ろ半分をソート

  //作業用配列に詰め直す
  for(int i = 0; i < n; i++){
    work[i] = array[i];
  }

  //マージ
  int left = 0;
  int right = n/2;
  for(int i = 0; i < n; i++){
    if(right == n){
      array[i] = work[left];
      left++;
    }else if(left == n/2){
      array[i] = work[right];
      right++;
    }else{
      if(work[left] < work[right]){
        array[i] = work[left];
        left++;
      }else{
        array[i] = work[right];
        right++;
      }
    }
```

```
  }

  return;
}
```

1.3.2 改良その 1

配列外参照を防ぐ為に、マージの際の条件分岐が複雑になってしまっている為、そこを改良しました。前半分を昇順、後ろ半分を降順にソートする事で、マージの際に配列の外を見る可能性が無くなり、条件が簡単になります。降順にソートをする場合は前半分を降順、後ろ半分を昇順にソートします。

```
#include<iostream>
using namespace std;

void MergeSort(int* begin, int* end);
void mergesort(int* array, int* work, int n);
void mergesort_rev(int* array,int* work,int n);

//半開区間
//配列確保の為に呼び出し
void MergeSort(int* begin, int* end){
  int n = end - begin;
  int *work = new int[n];
  mergesort(begin, work, n);
  return;
}

void mergesort(int* array, int* work, int n){
  if(n <= 1){
    return;
  }

  mergesort(array, work, n/2);
  mergesort_rev(array + n/2, work + n/2, n - n/2);//後ろ半分は降順にソート

  //データの詰め直し
  for(int i = 0; i < n; i++){
    work[i] = array[i];
  }

  //マージ
  int* end = work + n-1;
  while(work <= end){
    if(*work < *end){
      *array = *work;
      work++;
    }else{
      *array = *end;
      end--;
    }
    array++;
  }

  return;
}
```

```
//降順にソートする為に追加
void mergesort_rev(int* array,int* work,int n){
  if(n<=1){
    return;
  }

  mergesort_rev(array, work, n/2);
  mergesort(array + n/2, work + n/2, n - n/2);

  //データの詰め直し
  for(int i = 0; i < n; i++){
    work[i] = array[i];
  }

  //マージ
  int* end = work + n-1;
  while(work <= end){
    if(*work > *end){
      *array = *work;
      work++;
    }else{
      *array = *end;
      end--;
    }
    array++;
  }

  return;
}
```

1.3.3　改良その 2

array をソート→ work に詰め直す→ array にマージ

これはなんだか無駄な気がするので、ここを改良しました。

void mergesort(int array, int work,int n) の定義を、”work” を作業用配列としてソート後のデータを”array” に詰める、と解釈すると、mergesort(work,array,n/2) は”array” を作業用配列としてソート後のデータを”work” に詰める、事ができます。これにより、無駄な for 文を一つ消す事に成功しました。しかし、前処理として、work に array の中身をコピーする必要があります。

```
#include<iostream>
using namespace std;

void MergeSort(int* begin, int* end);
void mergesort(int* array, int* work, int n);
void mergesort_rev(int* array,int* work,int n);

//半開区間
//前処理の為に呼び出し
void MergeSort(int* begin, int* end){
  int n = end - begin;
  int *work = new int[n];
  for(int i = 0; i < n; i++){
    work[i] = begin[i];
  }
```

```
  mergesort(begin, work, n);
  return;
}

void mergesort(int* array, int* work, int n){
  if(n <= 1){
    return;
  }

  //呼び出し先でworkとarrayをクロス
  mergesort(work, array, n/2);
  mergesort_rev(work + n/2, array + n/2, n - n/2);

  //ここのfor文が丸々消える

  //マージ
  int* end = work + n-1;
  while(work <= end){
    if(*work < *end){
      *array = *work;
      work++;
    }else{
      *array = *end;
      end--;
    }
    array++;
  }

  return;
}

void mergesort_rev(int* array, int* work, int n){
  if(n <= 1){
    return;
  }

  //呼び出し先でworkとarrayをクロス
  mergesort_rev(work, array, n/2);
  mergesort(work + n/2, array + n/2, n - n/2);

  //ここのfor文が丸々消える

  //マージ
  int* end = work + n-1;
  while(work <= end){
    if(*work > *end){
      *array = *work;
      work++;
    }else{
      *array = *end;
      end--;
    }
    array++;
  }

  return;
}
```

1.4　終わりに

ソートを含め、プログラムの無駄な部分を省くチューニング作業は非常に面白いです。実はソートの関数呼び出しの際、スタックへの退避を減らす為に変数をなるべく少なくしています。また、最悪計算量が $O(N \log N)$ のマージソートは無駄を減らす事しかできませんが、最悪計算量が $O(N^2)$ のクイックソートはピボットの選び方の工夫で速度を速くする事ができます。これは面白い研究のテーマになり得ます。

何気に部誌を書くのは今回が初めてです。元部長なのに...。

第 2 章

このご時世でのサークル活動は制限だらけで苦しいと思ったらそうでもなかった件

terry

概要
コロナ禍でサークル活動が遠隔になったり制限があったりしたがためにサークルの加入をためらっているそこのあなたにむけて，サークルに入ってよかったことや実際どんな感じだったかをお伝えする記事.

2.1　そもそもサークルとは・・・？

我が電気通信大学はサークルをこのように紹介しています.[*1]

> 学業にいそしむことは学生の本分ですが、オンとオフをきっちりと分けることで、学生生活がより豊かなものとなります。オフの時間を彩る代表格にサークル活動があります。本学には体育系・文化系、同好会合わせて約 80 の大学公認団体があり、それぞれ多彩な活動を繰り広げています。毎年 11 月に行われる学園祭「調布祭」では各サークルが展示やコンサート、演劇、模擬店などで来場者を楽しませています。趣味が同じ仲間との時間は、きっとあなたにとってかけがえのないものとなるはずです。興味のあるサークルを見つけたら、ぜひその門を叩いてみてください。

自分は，サークルの活動はいろいろあれど，その活動がしたいという仲間が学年関係なく集まる場だと思ってます．縦のつながりを作るのはサークルが一番カンタンだと思っています．このつながりは役に立つことが結構あると思います.

[*1] 電通大ホームページより—https://www.uec.ac.jp/campus/extracurricular/club.html

2.2　コロナ禍の MMA

対面での活動がなくなってしまったのは大きいとは思います．ただ，他の運動部などと違って，オンラインでも活動に支障が出にくいというのは，不幸中の幸いだったかもしれません．オンラインでいろんな分野の勉強会，講習をやってもらいました．新入生視点だとそれだけに見えますが，今になって思えば裏のサーバーやシステム管理もやってたのだろうと思います．そのため活動は鈍った程度で，止まってはなかったのかなと感じています．これは仮入部だと見えないことですが，MMA の部内交流は盛んです．Slack というサービスを用いていろんな話題に関して話をしています．これはコロナ禍であろうと変わらなかったと思います．ただ，合宿がなくなったのは大きいですね．いつも打ち上げ花火を上げていたらしいのでそれが見れないというのは残念でしたね．自分はよく知らないんですよね... 自分のときにはなくなっていたので (悲しい)

2.3　サークル加入前の不安

これです．皆さん不安でしたよね．僕も不安です．ぶっちゃけ

サークル入る意味なくね？

と思ってました．理由は以下のとおりです．

- 入部することへのメリットが感じられなかった (MMA での活動は家で一人でもできるから)
- 月 500 円の部費を払いたくなかった．

ただ，こんなもんでした．強烈な理由はなかったんです．Twitter で知り合った MMA の先輩と ICPC というプログラミングの大会にチームで参加しました．その繋がりもあって，一回入ってみようかなぁと思いました．似たサークルの 1 つの工研さんは，半期で 6000 円[*2]らしいので，それよりは良心的ですし，一回入って先輩たちとのつながりを作ってみようと思いました．ただ，不安は不安で，入ったところでなにもできずに終わってしまうかもしれないと思ってました．

2.4　MMA 加入後の感情と行動と利益

ここでは MMA に入った後に，自分の不安がどう変化していったかということと，入った後になにをしたか，どんなメリットがあると感じたかを紹介しようと思います．

[*2] この金額は MMA の 1 年間の部費に相当する

2.4.1 MMA 加入後に感じたこと

MMA は 11 月から入部可能になりましたが，自分は少し遅れて 12 月に加入しました．最初はなにもできなくて，このサークルに部費を収めるだけのお財布状態でした．「やっぱりな」という気持ちでいっぱいでした．

結局無駄に金を払ってるだけじゃないか！

自分のこころの中の声はそう語りかけてきました．しかし，ここで終わるわけにはいかないという思いで，色んな場所への働きかけをはじめました．

同じお金を払ってるなら大学設備，教員，その他なんでも使い潰せ

という精神を持ってる自分はサークルもその中の一つに加えてしまうのでした．自分からガツガツ先輩方に食いついていって自分にとって良い方向に向くように活動をしてみました．色々行動を起こすと，こんなにいい環境なかなかないと思うようになりました．マジで入ってよかったと思います．

2.4.2 MMA に入ってやったこと

なにをやったかと言うと以下の通り．入って数ヶ月でいろんなことをやったと思います．12 月入ですよこの人w

1. 部員のみが見れる wiki や GitLab を片っ端から漁った．GitLab とは GitHub のような git のホスティングシステムです[*3]．
   - 先輩の ssh_config を発見し勉強した．技術を学ぶきっかけをもらった．
   - 過去の勉強会資料がそこらのネットを漁るより勉強になった．
   - 先輩による様々な情報 (部内限定公開) を知ることができた．これ有意義すぎた．
   - GitLab のおかげで過去に開発された kagisys[*4]などのソースコードが読めた．
2. Slack で質問をした．だいたいなんでも解決する．先輩方がつよすぎる．
3. 聞いた質問を解決出来なくても一緒に考えてくれる優しい先輩との繋がりを作った．
4. MMAOB が強い．社会人の方もいるため色々教えてもらった．
5. 部長になった．(入部 1 ヶ月でなにやってるんですか)
6. 新入生に対する部の対応の仕方の改革を動かすきっかけを作った．
7. 新入生講習 Plus という分野別講習会プロジェクトを立ち上げた．
8. GitLab pages を用いることで自分のブログサイトを立ち上げた．[*5]
9. サンドボックス環境として部のサーバー資源を借りて勉強をした．

[*3] これわからなくても入部すれば教えてもらえるよ

[*4] kagisys は部室の電子錠システム．これのおかげで部員がいつでも部室に入れる

[*5] こちら `http://terry.pages.gitlab.mma.club.uec.ac.jp/`

2.4.3　MMA に入ってよかったこと

なにがいいって技術勉強のモチベになりますね．いろんな分野の有識者がいるので，わからないことを聞ける環境が整ってていいです．あと他にもレンタルサーバーを借りないと出来ないことが無料で出来ちゃったりします．ただ，セキュリティの問題で学内 or 部内公開になったりします．これらすべてを合わせても部費月 500 円は安すぎると感じます．部内限定資料の充実さ，Slack での雑談にたまにためになることが流れてきたりと他にも特典が盛りだくさんです．

2.5　終わりに

MMA に入ってくれたら嬉しいけど伝えたいことはそこじゃなくて，**サークルでのつながりは皆様をきっと助けてくれる**でしょうということです．もちろん対面じゃないとしんどいサークルもあるので一概には言えませんが，サークルに入って活動をすることをおすすめします．入る前に**必ず**いろんなサークルを回ってみてください．その後で，うちがいいと思ったらぜひ入ってくださいね．MMA はオンラインでも問題なく活動できます．活動をしようというモチベがあれば，行動を起こせさえすればなんでもできるサークルなので．

そのときはぜひよろしくね．

ここまで読んでくださってありがとうございました．

第 3 章

VPS を始めた話・dotfiles を作った話

maccha

新年度から 3 学年となる、maccha と申します。この春休みに取り組んだ些細なことについて、述べます。

3.1　VPS を始めた話

3.1.1　はじめに

以前から、自分でサーバーを管理しながら Web サイトを公開したいと思っていた。自らサーバーを借りて、ドメインを取得し、Web サイトのソースコードも己で書き、それを公衆の面前で晒すことを考えたら、震えが止まらなかった。(ゾクゾク) この春休みに思い切って取り組んだので、そのことについて簡単に紹介したいと思う。

3.1.2　やったこと

VPS を借りる

レンタルサーバーは、ConoHa(https://www.conoha.jp/) の VPS を借りることにした。選んだ理由は比較的安いことと、マスコットキャラクター (?) の美雲このはちゃんがかわいいことである。ConoHa のアカウントを作成した後、VPS を新規契約した。割り振られていた IP アドレスで、自機から ssh 接続をした。その後は ConoHa の Web サイトにあるガイダンスに従って、初期設定などを進めた。もちろん、VPS もドメインも有料である。

ドメインの取得

こちらも ConoHa から購入することにした。念願の自分だけのオリジナルドメインをついに購入することが叶った。この時初めて知ったが、ドメインの後ろの部分 (TLD:トップレベルドメイン) は価格がバラバラのようだ。その後、DNS サーバーの設定をした。

Web サイト公開

Web サイトのデザインは、BootStrap を使ってすぐに完成させた。Bootstrap は統一感のあるデザインを実現しやすく、また使いやすい上、機能が豊富なのでおすすめである。

サーバーに Apache をインストールし、色々初期設定を進め、ポートも開放した。これにて、Web サイトが公開された。

サーバー証明書

ついでにサーバー証明書を発行したいと思ったので、Let's Encrypt による証明書を発行し、設置した。発行から 90 日が経過すると使えなくなるため、定期的に更新することを忘れないようにしたい。

3.1.3 今後やりたいこと

今後 VPS を使って、さらにメールの転送用サーバーや VPN、ファイルサーバーを構築したいと思っている。

3.1.4 補題: 美雲このはちゃんについて

美雲このはとは、ConoHa のマスコットキャラクター (多分) である。清楚で可愛いという設定らしい。(実際可愛い)CV は上坂すみれさん。Web サイトを訪れると、このキャラのグッズや CD が販売されており、マンガも連載されているようだ。Twitter もやっており、更新頻度も高い方だと思う。また、このはちゃんには YouTube チャンネルがある。最近は 3D モデルがデビューし、マイクラ実況が投稿されている。(VTuber かな？) 大人の都合上、中の人は上坂すみれさんではなく、ConoHa スタッフである。(声はかわいい) ということは、中の人は、勤務時間中にマインクラフトをプレイしていることになるのだろうか。

3.2 dotfiles を作った話

3.2.1 はじめに

ホームディレクトリにある設定ファイルを、読者諸君らはどのように管理しているのだろうか。悪戦苦闘の末出来上がった最高の vim の設定ファイルや、快適な開発ライフを送るための Git の設定ファイルを自機に有している人も少なくないだろう。そんな素晴らしい環境が、自機以外でも構築できたら？すなわち、外部のサーバーをいじる機会がある人 (特に電通大生なら sol や sed) が、その環境でも自機と同じ作業環境を容易に展開できたら、便利ではないだろうか。さらに自機を新しいマシンにする際も、これまで通りの環境をすぐに用意できれば、非常に楽ではないだろうか。このような願望がある方は、ぜひ設定ファイルを管理する Git リポジトリを作ってみてはどうだろうか。Git 上で設定ファイルを保管し、必要に応じてそのリポジトリを clone または pull して、各種設定ファイルをホームディレクトリなどに設置

してしまえば、いつでも (インターネットが利用できる環境下において) 使い慣れた作業環境を構築できる。このようなリポジトリ (ディレクトリを指すこともある) を、dotfiles という。

3.2.2 やったこと

GitHub 上で dotfiles リポジトリを作り、自機のホームディレクトリに clone した。そして、作成された dotfiles ディレクトリに、各種設定ファイルを置いた。後で GitHub で公開することを踏まえ、.gitconfig に書かれている user 名とメールアドレスは削除し、また.ssh ファイルは dotfiles で管理しないようにした。

```
~
├── dotfiles
│   ├── .vim
│   │   └── color
│   │       └── hybrid.vim
│   ├── .gitconfig
│   ├── .vimrc
│   └── install.sh
└── 各種設定ファイル
```

さらに、install.sh を作成した。これは、dotfiles 内の設定ファイルに到達するシンボリックリンクを、ホームディレクトリに置く指示を書いた sh ファイルである。他の環境で dotfiles を clone した後に実行して、設定ファイルのセッティングをするようになっている。

以上が完了したら、GitHub に push した。試しに借りている VPS 上で、dotfiles リポジトリを clone して使ってみたが、自機と同じような vim の操作ができるようになっていた。

3.2.3 今後やりたいこと

時間がある時に、さらに dotfiles をアップデートしていきたいと思っている。特に、vim や Git の設定の改善などである。

vim や Git の他にも bash 関係の設定をはじめ、様々な設定ファイルを一元的に管理できる。読者諸君らも dotfiles を始めてみてはどうだろうか。

第 4 章

初級モデリング入門

Nasatame

4.1　はじめに・環境構築

この記事は、Unity でゲームを作り始めたが自分が思い描く 3D 素材が見つからない、Asset Store の素材に物足りなさを感じてしまう、そういった方へ向けて Unity 上で簡単な 3D 素材の作り方を伝えるということを目的としています。特にこの章では、環境構築について取り扱います。

今回この記事で使用するツールは以下の通りです。

- Unity 2019.4.3f1
- ProBuilder 4.2.3
- ProGrids 3.0.3

Unity は公式サイトからダウンロードすることが出来ます。インストールについては、公式サイトを確認してください [1]。

ProBuilder と ProGrids については、Unity 内から Package Manager を用いてインポートしてください（図 4.1）。　バージョンは、現在使用している Unity で推奨されているものを選択してください。バージョンによって ProBuilder、ProGrids の UI が多少変更されている可能性がありますが、適宜読み替えてください。また、ProBuilder には、機能のボタンをアイコンで表現する機能がありますが、記事中ではアイコンから意図が分かりづらいため文字表記を用いています（図 4.2）。

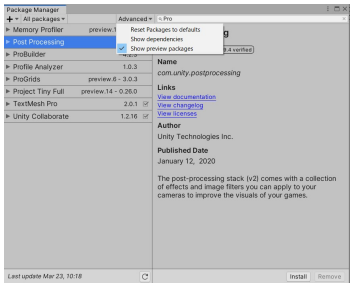

図 4.1　Package Manager

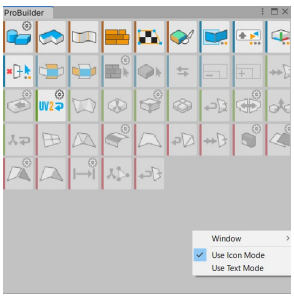

図 4.2　Icon 表示

4.2　概形の作成

この章では、モデリングにより素材の概形を作る方法について取り扱います。この記事では、サンプルとして MMA の Logo(図 4.3）を作ります。

4.2.1　NewPolyShape 機能を用いて概形を作る

Unity 上でモデリングをするためのパッケージである ProBuilder の機能を使って概形を作ります。ProBuilder は **Tools/ProBuilder/Probuilder-Window** から出せるパネルで操作します。パネルは、選択している対象が、なにもない、オブジェクト、辺、頂点で出来る操作が変化するため注意してください。

NewPolyShape 機能は厚みが一定であれば、大抵の図形の形を作れる機能です。穴は開けられないため、この機能を用いて A の穴の開いていない形と M の概形を作ります (図 4.4)。

図 4.3　目標とする MMA Logo

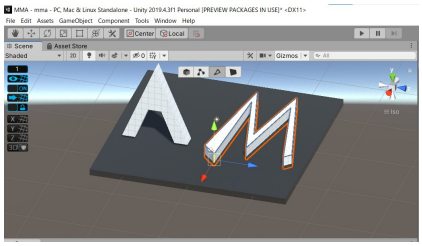

図 4.4　概形

4.2.2　概形の調整

ここからは ProBuilder の図形を指定したマスのグリッドに乗せる機能を持つ、ProGrids を使用します (図 4.5)。図の一番上は、グリッドの大きさ、3 番目に機能の **ON/OFF** があります。ON にした上で、ProBuilder で辺を選択します。そして、通常のオブジェクトの移動と同じように移動させ、形を整えると図 4.6 のようになります。

図 4.5　ProGridsWindow

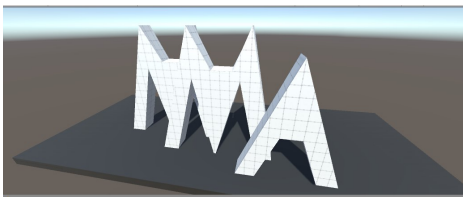

図 4.6　調整後の概形

4.3　テクスチャによる見た目の調整

この章では、マテリアルをテクスチャから作成し、それを貼り付けることでモデルの見た目を調整する方法について取り扱います。

4.3.1　テクスチャの作成

テクスチャの作成は、Power Point などで行います。一つのマテリアルは以下の 5 つのテクスチャから成り立ちます [2]。

Albedo　表面の色を決める。カラー画像
Metalic　白い部分はより金属らしい質感になる。モノクロ画像
Normal　表面の凹凸を表現する。モノクロ画像を Unity 上で変換する
Emission　光の当たり具合を表現する。白い部分がより光が当たっている。モノクロ画像
Occlusion　オブジェクトの発光具合を表現する。その色で発光する。カラー画像

形式は png でテクスチャの作成は各自行ってください。また、Albedo と Metalic は透過させたい部分は透明度を指定しなければなりません。今回であれば、A の穴の部分は Alpha=0 にしました (図 4.7)。

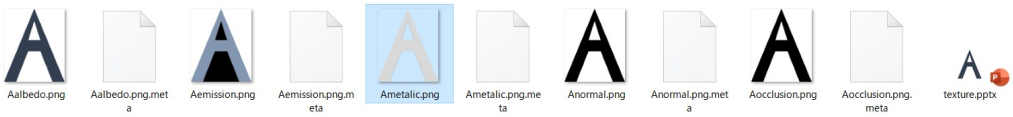

図 4.7　テクスチャ

4.3.2　マテリアルの作成

マテリアルの作成のため、Normal テクスチャを特殊な形式に変換します。TextureType を Normal map にし Apply することで図 4.8 のように変換します。最後に Material を作り、図 4.9 のように画像をドラッグし、Rendaring Mode 設定することで完了です。

図 4.8　法線画像への変換

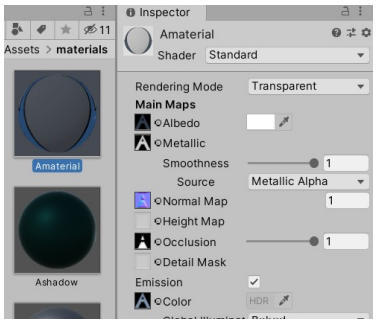

図 4.9　マテリアルの設定

4.3.3　マテリアルの貼り付け

マテリアルの貼り付けは、ProBuilder で面を選択した上で MaterialPallet で **Alt ＋ 1** などを押すことで出来ます。貼りたいマテリアルは、ドラッグ＆ドロップなどで前もって設定しておいてください (図 4.10)。
また、図 4.11 のように表示がずれることもあると思います。その際は、UVEditor で調整してください (図 4.12)。また、透過するのは面だけなので A の反対側の面は消去しておいてください。

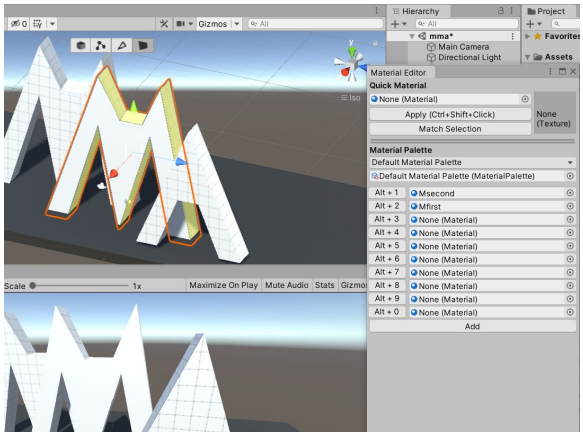

図 4.10　Material Editor

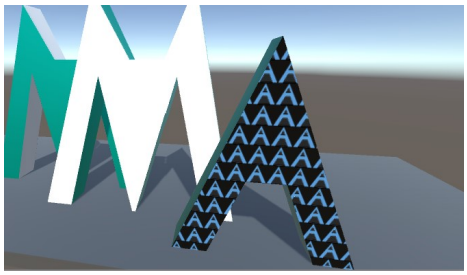

図 4.11　表示ずれ

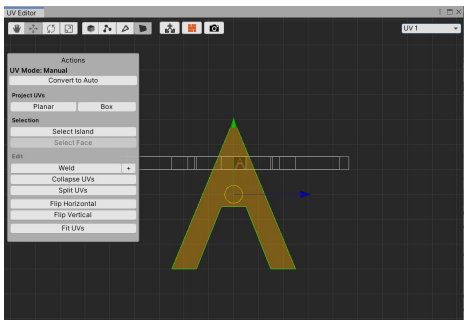

図 4.12　UVEditor

4.4　おわりに

最終的に色々と調整したものが、図 4.13 です。ロゴにしっかりと似ていると思います。この記事では、簡単な 3D 素材の制作方法について、モデリング、テクスチャの作成方法とその貼り付け方という手順で説明を行いました。もし質問等がございましたら Nasatame[*1] までお願いします。

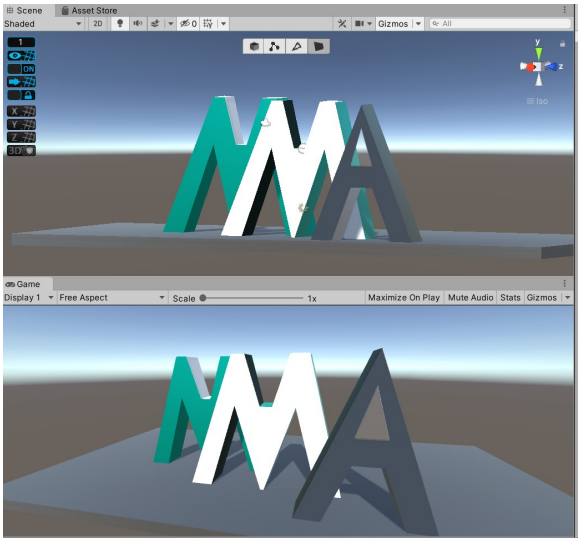

図 4.13　完成したロゴ

参考文献

[1] https://unity.com/ , Unity Technologies, March.2021.

[2] https://mslgt.hatenablog.com/entry/2015/10/05/013746/ ,【Unity】Standard shader に触ってみた, vaguely, March.2021.

[*1] s1810741@edu.cc.uec.ac.jp

百萬石 2021 -春- © 電気通信大学 MMA

2021 年 4 月 10 日　初版第一刷発行　【本書の無断転載を禁ず】

著　者　tonphy, terry, maccha, Nasatame
表　紙　tonphy
編集者　tonphy
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


NETGEAR
NETGEAR
XS724EM
Combo Port
NETGEAR
Power
Fan
Factory Defaults
208
318
901
110
106
602