1　実行方法

1.1　動作環境

Linux のディストリビューションのひとつである Ubuntu で動作を確認している．Ubuntu で動作させるための必要なアプリケーションは以下の通りである．

```
sudo apt-get install build-essential
sudo apt-get install cmake
sudo apt-get install meshlab
```

コンパイル時に必要なビルドシステムは build-essential と cmake が必要であり，生成された stl のデータ確認のため meshlab というアプリケーションを用いている．

Visual Studio でのコンパイルも可能であるが，その場合 Struct.h の最初の記述を変更する．

```
#define _CRT_SECURE_NO_DEPRECATE
#include <stdio.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES
#include <math.h>
```

1.2　ファイル構造

この zip の中のファイル構造を示す．

```
.
├── CMakeLists.txt
├── Struct.h
├── document
│   └── document.pdf
├── kadai.h
├── kadai1A.cpp
├── kadai1A.h
├── kadai2A.cpp
├── kadai2A.h
├── main.cpp
└── plot
    ├── Tetra-Triprism.stl
    ├── Tetra01.stl
    ├── Triprism01.stl
    └── tetra_point.dat
```

tetra_point.dat が各立体の頂点の初期位置が格納されたデータであり，Tetra01.stl と Triprism01.stl が課題 1 の結果，Tetra-Triprism.stl が課題 2 の結果である．

1.3　コンパイル方法と実行方法

```
mkdir build
cd build
cmake ..
make
```

```
./実行ファイル名
```

2　実装した関数の説明

2.1　課題 1

指定された点から四面体と三角柱を作成するために実装した関数を示す.

```
void InputDatFile(POINT p[], char *fname, double loop);
```

- 立体用データを外部ファイルから読み込み配列 p に格納する.

```
void set_tetra(double vs[][VEC_SIZE]);
```

- データから立体を表す座標を 2 次元配列に格納 (四面体用)

```
void set_triprism(double vs[][VEC_SIZE], double h);
```

- データから立体を表す座標を 2 次元配列に格納 (三角柱用)

```
void norm_triangle(double n[VEC_SIZE], double v0[VEC_SIZE], double v1[VEC_SIZE], double
    v2[VEC_SIZE]);
```

- 外向き法線ベクトル作成する
- 法線ベクトルを求めた結果は配列 n に格納される

```
void prStlProlog(char *label, FILE *fp);
```

- STL 形式の開始行を作成

```
void prStlEpilog(char *label, FILE *fp);
```

- STL 形式の終了行を作成

```
void prStlFacet(double n[VEC_SIZE], double v0[VEC_SIZE], double v1[VEC_SIZE], double v2[
    VEC_SIZE], FILE *fp);
```

- STL 形式の三角形パッチを作成
- 頂点と外向き法線ベクトルを引数に

```
void CombinationTetra(double vs[][VEC_SIZE], FILE *fp);
```

- 外向き法線方向に右ねじが進む方向の点の組み合わせ作成 (四面体用)

```
void prStlTetra(double vs[][VEC_SIZE], char *label);
```

- STL 形式の四面体作成

```
void CombinationTriprism(double vs[][VEC_SIZE], FILE *fp);
```

- 外向き法線方向に右ねじが進む方向の点の組み合わせ作成 (三角柱用)
- 外向き法線方向に右ねじが進む順番になるようすべて列挙している

```
void prStlTriprism(double vs[][VEC_SIZE], char *label);
```

- STL 形式の三角柱作成

```
void stlb_k01(double tt[TETRA_V_NUM][VEC_SIZE], double tp[TRIPRISM_V_NUM][VEC_SIZE]);
```

- 四面体と三角柱を STL 形式でそれぞれ作成 (課題 2A の 1)

2.2 課題 2

```
double GetRandom(double min,double max, int digit);
```

- 範囲と所望の小数点以下の桁数を指定して乱数を得る

```
AXIS randCoordinateAxis();
```

- 任意の原点位置と任意の軸方向を持つ新たな座標系をランダムに作成
- 課題 1B で作成したものと同じ原理で結果を得ている

```
void CalcRotationMat(POINT input[], POINT output[], double rotation_mat[][VEC_SIZE], int
    loop);
```

- 引数にとった変換行列を立体を表現している座標群に施す

```
void TransformPosition(char *label, FILE *fp);
```

- 取る引数 (label) によって欲しい立体 (四面体か三角柱用) をランダムな座標系で作成

```
void TransformPositionAll(char *label);
```

- 課題 2A の 2 を実行する

3　結果

それぞれの課題で結果を示す．

3.1　課題 1

四面体を出力した結果を Fig. 1，三角柱を出力した結果を Fig. 2 に示す．

3.2　課題 2

$O_1X_1Y_1Z_1$ に四面体，$O_2X_2Y_2Z_2$ に三角柱を，各任意座標系における初期位置に配置した結果を Fig. 3 に示す．

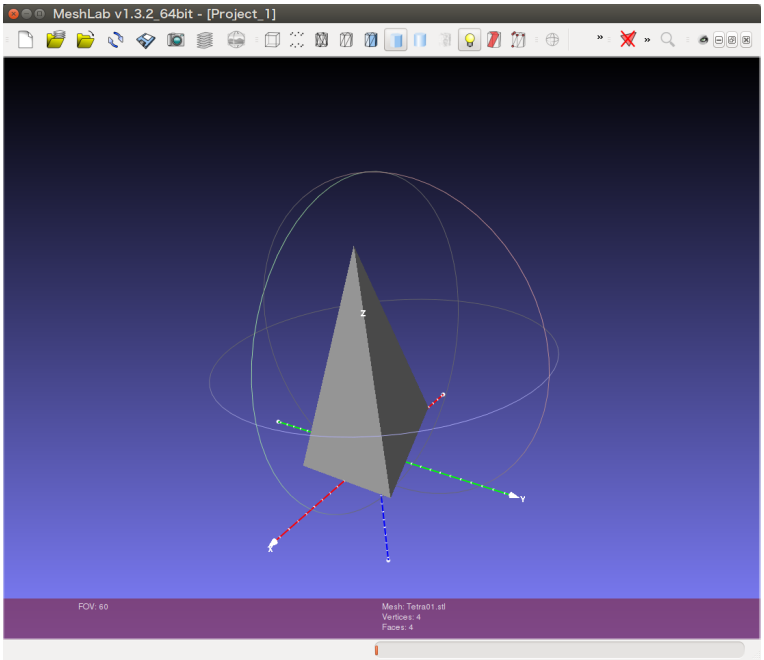

Fig.1　四面体を作成した結果

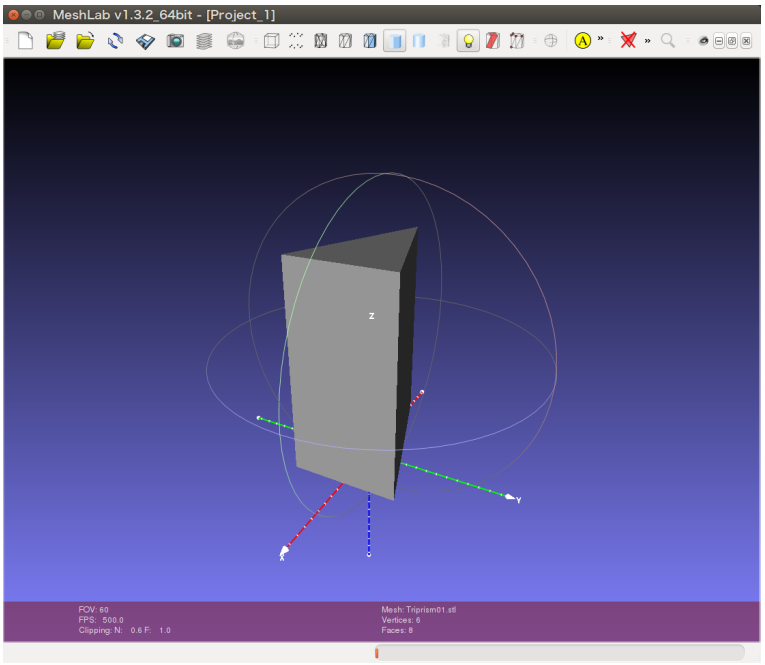

Fig.2　三角柱を作成した結果

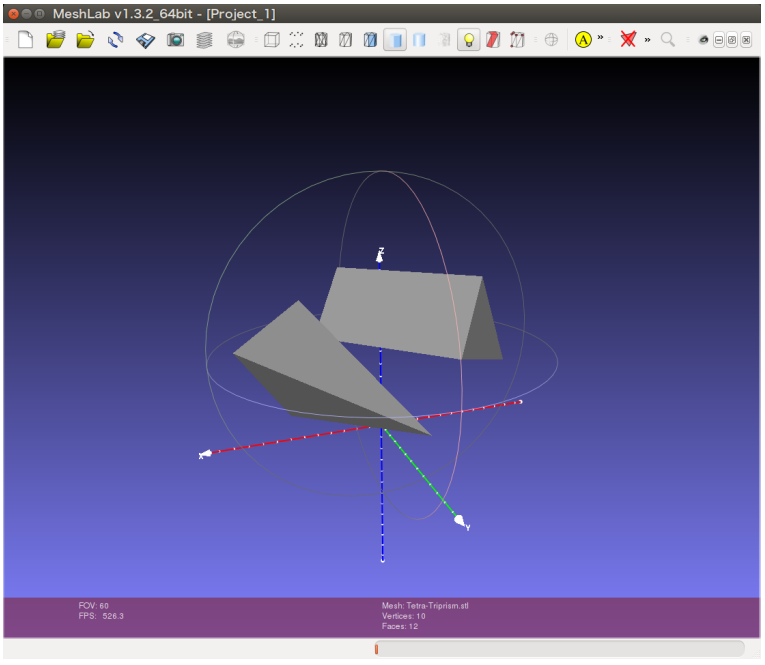

Fig.3　四面体と三角柱を任意の座標系の初期位置に配置した結果