

グラフィックス演習（第4週目）

日付：

学生番号

氏名

● グループの学生のリスト（参加者のみ）：自分自身の担当については詳細に、他のグループ員の分に関しては、最小限で良い。リーダー、サブリーダー、書記に関しては少なくとも明記せよ。

	ファースト ネーム	役割	変更があった役割分担について記述すること
1.本人			
2			
3			
4			
5			

【自由記述】グループでの活動状況に関して配慮した点や苦慮した点を記述せよ。

● 本日の課題の評価

項目	評価	1 Excellent	2 Good	3 Fair	4 Poor
要件定義書		各グループの要件定義書の問題点を洗い出し、次回以降の活動が出来るように修正できた。	各グループの要件定義書の問題点を洗い出すことができた。	要件定義書の内容を理解し、自分の担当が理解できた。	要件定義書の内容を理解できなかった。
開発予定の企画		今後の開発するべき C 言語のコードを見通すことが出来、開発時間、開発の目処がたった。	今後の開発するべき C 言語のコードが想定できた。	今後の開発するべきソースコードのアクティビティ図を理解する事ができた。	今後の開発するべきコードが見通せていない。
グループでの活動		グループで活動することにより、自分だけでは洗い出せなかった問題を明確化できた。	グループで問題点を出し合い、個別の活動に活かした。	自分自身が分からない点をグループ員から明らかにすることができた。	グループでの意見交換はあまり活発ではなかった。
最終成果物の理解		分子グラフィックスを自由に作成できる	OpenGL の理解を通して、分子グラフィックスの設計ができる。	目標とするプログラムが理解できた。	何を作成するべきかが理解できていない。

● 前回からのグループ演習についてのアンケートに答えて下さい。

1. GITを利用して活動する環境について下記の語句を選択した後、自由記述に記して下さい。

意見交換は容易である                      意見交換は可能である                      難しい

(自由記述)

2. 本日のグループ型の演習についての意見を、下記の語句を選択した後、自由記述に述べて下さい。

学習する上で有効                      有効性を感じない                      未だよく分からない

(自由記述)

その他、本グラフィックス演習に関する意見・質問、要望があれば記述して下さい。

本日の課題まとめ（団体）：4 週目－ （共通変数）

日付： グループ（番号）：

開発コード名（プロジェクト名）：

リーダー： （自署）

サブリーダー： （自署）

書記： （自署）

機能要件に階層性がある場合には、枝番をつけてもよい。

#### 要件定義書の報告

要件 番号	機能要件	実現方法	実装 状態	担当
1	各コールバック 関数間で共有す る変数の宣言	<p>（必要となる外部変数のリスト）</p> <p><code>extern int animationFlag</code> （アニメーションさせるかどうかに関するフラグ変数。外部変数として宣言、1 でアニメーション、0 でストップ、<code>config.h</code> に <code>extern</code> 宣言、<code>main.c</code> に実体を宣言する。）</p> <p><code>extern int displayFlag</code> （どの表示方法を選択するかに関するフラグ変数。外部変数として宣言。1, 2 で画像表示方法を変更する。<code>config.h</code> に <code>extern</code> 宣言、<code>main.c</code> に実体を宣言する。）</p> <p><code>extern PDB pdb;</code> （実際に利用する PDB のデータを格納しておく為の変数。<code>config.h</code> にて外部変数にて宣言。<code>main</code> 関数で <code>pdbRead</code> で読み込み、<code>display</code> 関数等で利用する。<code>config.h</code> に <code>extern</code> 宣言、<code>main.c</code> に実体を宣言する。）</p>	<p>未実装</p> <p>未実装</p>	

--	--	--	--	--

本日の課題まとめ（団体）：4 週目－ （共通 API）

日付：

グループ（番号）：

開発コード名（プロジェクト名）：

リーダー：（自署）

サブリーダー：（自署）

書記：（自署）

機能要件に階層性がある場合には、枝番をつけてもよい。

要件 番号	機能要件	実現方法	実装 状態	担当
1	PDB ファイルを読み込みコールバック関数、OpenGL の環境などを設定する。	main(int argc, char* argv[]) 第一引数として、PDB ファイル名を設定し、main 関数内部で外部変数の pdb に対して読み込む事とする。	未実装	
2	キーボードが押されたときの振る舞いを定義 q, Q で終了。s でアニメーション。	key(unsigned c, int x, int y) q, Q では、exit 関数を呼び出す。 s では、animationFlag 変数の 0 と 1 を交互に変更する。 1, 2 等の 0 - 9 の数字キーの入力により、displayFlag 変数を 1, 2 と切り替える。	未実装	


--	--	--	--	--

#### 第4週目 本日の課題

作業：【団体戦】分子グラフィックスについての要件定義書の作成

機能、実装方法などの検討、アクティビティ図によるアルゴリズム表示 ●本日

の作業：要件定義書の作成と分担

1. 今回のプロジェクト（分子グラフィックス：簡易 RASMOL）のリーダー、サブリーダーを決定せよ。各会議では、書記を決め、議事録を作成すること。
  - (ア) リーダーの仕事は、プロジェクト全体が動くための仕組み作りと工程管理にある。  
Gitのレポジトリを作成し、ソースコードの管理を行う事。
  - (イ) サブリーダーの仕事は、リーダーの補佐であり、作業の流れを確認する。
2. 開発する分子グラフィックスソフトの名前、もしくは、開発コード名を決定せよ。
2. 作成する分子グラフィックスの機能について、共通機能に加えて、Rasmol や chimera の機能から想定（同じでなくてもよい）し、リストアップせよ。
  - (ア) 共通に要求する機能に関しては、リストに加えること。
  - (イ) 添付の要件定義書（本日、1部提出）を利用しながら、実装する、もしくは、実装したい要件を洗い出す事。
  - (ウ) 提出までに作成可能な機能のみならず、あると望ましい機能についてもリストし、今回は作成しない旨記述せよ。
  - (エ) 要件定義書に関しても全体で共有し、作成・提出せよ。
3. 上記で列挙した機能を分類し、それぞれの機能を関数に割り当て、関数毎に、担当を 分担する。
  - (ア) 1月24日までにそれぞれの機能に対して、アクティビティ図を作成し、プロトタイプを作成するための、分担せよ。次回までにアクティビティ図を作成し、提出する。
  - (イ) 各人が担当となる機能（関数）として2つ以上を担当すること。
  - (ウ) 最終的に、作成する担当とは異なっても良いし、複数人の開発を行ってもよい。もちろん、担当者が同じであるほうが対応しやすい。
  - (エ) 実際にプロジェクトが進行中である中で、途中の変更や複数分担もよしとする。その旨、最終的に提出する要件定義書等にその旨を記述すること。
  - (オ) 毎回集まった際に、ここで作成した要件定義書を整理して提出してもらう。

5. 各々の機能の実装を検討する中で、継続的に必要となる共通変数の洗い出しを行ってもらいが、予め分かるものに関してはその旨、定義しておく。必要となる外部変数以外にも、想定されるものはリストアップしておくこと。

共通変数に関しては【要件1】にそれを記述する。正式なリストアップは、アクティビティ図を作成する中で、必要となる情報を洗い出す作業を通して実施する。

【次回までの作業】1月24日までに実施し、提出すること

【グループ提出】グループ毎に、本日提出した要件定義書を整理し、その後のアクティビティ図の作成、プロトタイプソースコードの作成を行い、提出せよ。

- 次回、その要件定義書を打ち出したものを持参すること。

【個人提出】担当する要件に対して、ソースコードに関するものは実装方法に関して、それぞれの担当の関数毎にアクティビティ図として表現し、提出せよ。プロトタイプソースコードの作成も行ってみよう。

関数毎にA4——1枚のアクティビティ図となるように表現し、提出せよ。その際、実装に必要な時間を想定し、記述せよ。また、共通で必要となる変数もリストアップし、グループで整理できるようにせよ。

- 次回、担当した関数のアクティビティ図を印刷し、持ち寄ること。

次回、上記の提出物を元に、必要な外部変数、引数などの摺り合わせを行い、要件定義書を整理する。

---

#### 参考ソースコード

下記に示したものは参考になるかもしれないソースコードである。ただ、この中では原子位置が配列で表現されている。その点をこれまで作成してきたリスト構造に変更する必要がある。

```
$ git clone http://edu-git.bio.kyutech.ac.jp/gitbucket/git/tacyas/rasmolTest.git
$ cd rasmolTest
$ make depend
$ make
$ ./myTest
```



●キーボード（要件定義書：機能2を参照）

終了： q または、Q による終了

アニメーション（y軸回転のロック（±30度の範囲で行行き来する。））：

s による切替（トグルスイッチ）

表示方法の切替： 1、2

キーボードにより、下記の表示内容の2種が切り替えられること

再描画コールバック関数を切り替えるか、若しくは、再描画コールバック関数内で switch 文で切り替えることにより実現できる。

●マウス

分子の回転： Rasmol の左ボタンと同様の機能

分子の移動： Rasmol の中ボタンと同様の機能

分子のズーム： Rasmol の Shift+左ボタンと同様の機能

z軸周りの回転： Rasmol の Shift+右ボタンと同様の機能

●表示

透視投影に対応する事。

ダブルバッファ、フルカラー、背面面削除（デプスキュー）に対応する事。

平行ライト一つ（固定、移動はいずれでもよい）以上指定し、計算できること。

物質の色を指定し、その計算できること。

シェーディングの計算（フラットシェーディング）ができること。

●表示内容

タンパク質分子の情報をPDBファイルから読み込んできた後、最低、下記に示す2つ以上の異なる表示方法にて実現すること。読み込むファイル名は、起動時に、コマンドの第一引数として記述し、main関数内部で読み込む事

・起動方法： \$ コマンド名（各グループ設定） PDBファイル名

・表示方法1： 全ての原子を球：色は元素種

C：灰色、O：赤、N：青、H：白、P：黄色、その他：黄緑

・表示方法2：α炭素を線で結んだ表示が可能であること：色はアミノ酸種

酸性アミノ酸：赤、塩基性アミノ酸：青、極性アミノ酸：黄緑 疎水性アミノ酸：灰色

※起動時、読み込み時には、分子全体が視錐台に入り、全体が表示できること。そのためには、次を満たす必要がある。

視点を分子の中心、分子の大きさから、分子中心とカメラ位置との距離を計算する必要がある。

● 拡張機能の例（拡張機能に関しては、加対象とする）

下記、以外の機能拡張でも問題ない。自由に設定してよい。

ステレオ眼鏡への対応：Quad buffer を設定する

表示内容

色の変更（青から赤への変化）：カラーマップを変更する必要がある(myTest)参考

温度因子（低→高）

占有率（低→高）

N末端からC末端への変化

重心からの距離

アミノ酸毎にすべて色を変える

原子の大きさの変更

可能であれば、vdw半径に応じたもの

密度表示への対応

pdb2dsn6 を参考にすると計算可能である。

背景の指定

背景をコマンドラインから設定する

マウス

マウスで指した原子の情報を画面、もしくは、コンソール上に記述する

Rasmol のラベル機能に対応

キーボード

色の切替

特定のキーボードを押すと、コンソールからの1行入力を受け付ける

たとえば、PDBファイルの変更、背景の変更など

● 関連している技術要素(myTest を参考にせよ)

**異なるファイルの関数間での変数の受け渡し**（関数での引数で指定できない場合）

外部変数（グローバル変数）で指定(extern 宣言)

個別に宣言

外部変数を構造体で宣言し、構造体で指定される変数を外部変数で指定

宣言する変数を集合で取り扱うことができる。

外部変数（グローバル変数）を利用したくない場合

主として必要なファイルで、外部変数（ただし、static 宣言でスコープをファイル内に閉じる）、もしくは、静的変数（関数内部で static 宣言）を設定し、それらの変数のポインタを返す。

PDB ファイルの読み込み

main 関数において、ファイル名、もしくは、PDB構造体からなる変数を宣言し、上述の方法で変数を受け渡す。

### 高速な表示

OpenGL において、高速表示を行行う為には、displaylist と呼ばれる機能を用いる方法が一般的である。

1 回目目に基本的な表示関数をコンパイルし、2 回目目以降は呼び出すのみで表示できる仕組み。

### 視錐台の指定方法

gluPerspective/gluLookAt あるいは、glFrustum/gluLookAt で指定する必要がある。

myTest では、カメラの位置を自分で動かして設定している

myDisplay.c/myCamera.c を参考

glutSample では、物体そのものを回す形で実現している。

可能であれば、それらは別に指定しておく、便利である。

ただし、ライトの位置も意識する必要がある点に注意が必要である。

どこで呼び出すかにより、振る舞いが異なる