電界中の電子の運動
シミュレータ作成

精密工学科プログラミング基礎
資料

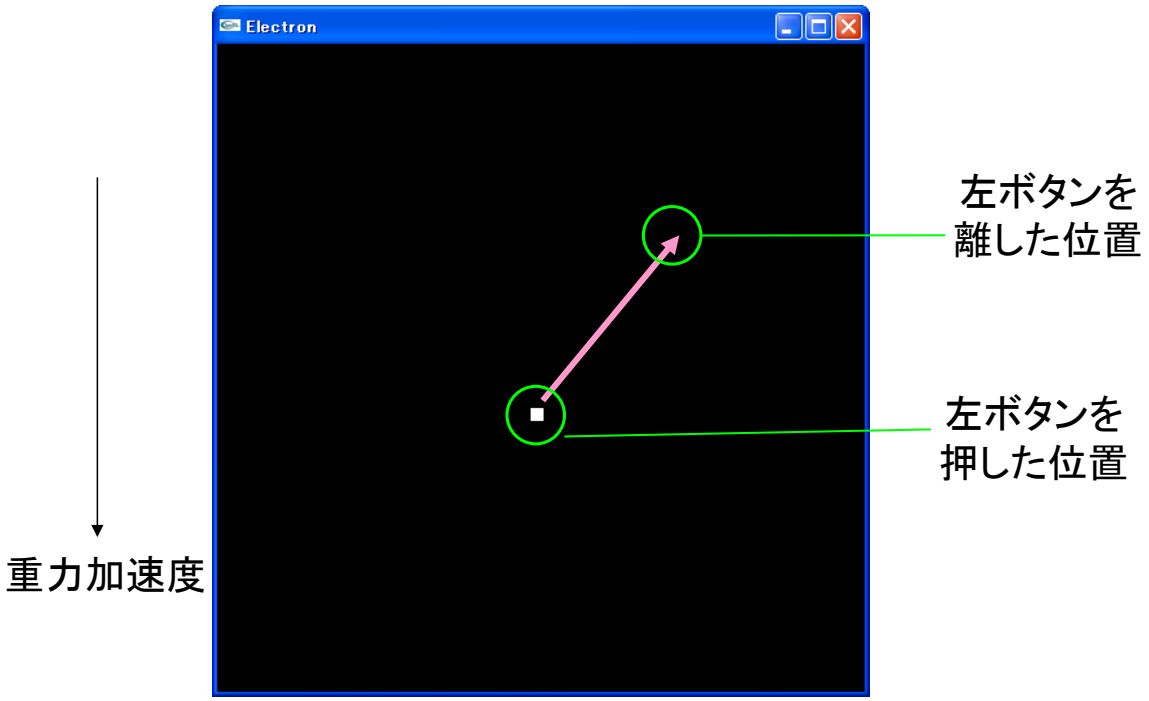
課題の目標

- 電子の運動をシミュレーションするプログラムを，
  サンプルプログラムを基にして作成する.

- 課題を通じて学べることは以下の通り.
  - 2次元グラフィックス
  - マウス入力の処理
  - 常微分方程式を計算機で解く方法
  - 電磁気学の復習

サンプルの実行結果

- 点の落下運動をシミュレーションする
  - マウス操作により初期位置と速度を指定する



コンパイル方法

- OpenGL と GLUT というライブラリを使用

```
gcc -framework OpenGL -framework GLUT sample.c
```

MAC のコンパイルにおいて、
ライブラリを指定するオプションです.

メモ :
OpenGL : 2D・3Dグラフィックス描画のためのライブラリです.
GLUT : ウィンドウを出したりマウスの操作を行うための OpenGL の補助ライブラリです.

プログラムの概要

プログラム全体で使う変数 (広域変数)
- 点の位置と速度，マウスが押された位置

画面の更新が必要な時に呼ばれる → display 関数
- 点の描画 → 6ページ

定期的に繰り返し実行される → idle 関数
- 時間経過による，点の位置・速度の更新 → 8ページ

マウスが押されたとき実行される → mouse 関数
- 点の初期位置・速度の設定 → 7ページ

今回は変更の必要なし

reshape 関数
- ウィンドウサイズに関わるもろもろの処理

main 関数
- ウィンドウの初期化，GLUTへの関数の登録など

点の描画 (display 関数)

- 色と大きさ・座標を指定して点を描く
  - “gl” で始まる関数を呼び出す

例：10ピクセルの大きさの紫の点を(100,100) と (50,20) に2点描く

```
glPointSize(10);
glColor3f(1, 0, 1);

glBegin(GL_POINTS);
  glVertex2d(100.0, 10.0);
  glVertex2d(50.0, 20.0);
glEnd();
```

色の指定は (赤，緑，青)を 0～1の float で設定

Begin と End で囲んで Vertex で座標を指定 (2d は 2次元の double という意味)

メモ：
その他に線分・多角形を描くことが出来ます．
- glBegin の引数を GL_LINES に変えると線分 (2点指定で1線分)
- glBegin の引数を GL_POLYGON に変えると凸多角形 (n 角形は n 個の点を指定)

曲線を描きたいときは短い線分を沢山描きます．

マウスボタン操作の処理関数

- 引数は以下の通り
  - ボタンの種類 (GLUT_LEFT, GLUT_MIDDLE, GLUT_RIGHT)
  - ボタンの状態 (GLUT_DOWN, GLUT_UP)
  - ボタンの押された座標 (左上を原点とする座標．単位はピクセルで int 型)

```
void mouse( int button, int state, int x, int y ){
  if( button == GLUT_LEFT ){
    if( state == GLUT_DOWN ){
      /* 左ボタンが押されたときの処理 */
    }
    else{
      /* 左ボタンが離されたときの処理 */
    }
  }
  else if( button == GLUT_RIGHT ){
    /* 右ボタンに関する処理 */
  }
}
```


微分方程式の数値解法 (idel 関数の中)

- 微小時間 dt において，
  微分値が一定であると仮定し、計算を行う．

```
/* 速度による位置の変更 */
  p[0] += dt*v[0];
  p[1] += dt*v[1];
/* 加速度による速度の変更 */
  v[0] += dt*a[0];
  v[1] += dt*a[1];
```

※ サンプルの
加速度ベクトルは
$\mathbf{a}=(0, 9.8)$

2階常微分方程式

$$\begin{cases} \dfrac{d\mathbf{p}}{dt} = \mathbf{v} \\ \dfrac{d\mathbf{v}}{dt} = \mathbf{a} \end{cases}$$

微分を1次近似

更新後 $t+\Delta t$，更新前 t

$$\begin{cases} \dfrac{\mathbf{p}^{t+\Delta t} - \mathbf{p}^{t}}{\Delta t} = \mathbf{v}^{t} \\ \dfrac{\mathbf{v}^{t+\Delta t} - \mathbf{v}^{t}}{\Delta t} = \mathbf{a}^{t} \end{cases}$$

プログラム化

$$\begin{cases} \mathbf{p}^{t+\Delta t} = \mathbf{p}^{t} + \Delta t\, \mathbf{v}^{t} \\ \mathbf{v}^{t+\Delta t} = \mathbf{v}^{t} + \Delta t\, \mathbf{a}^{t} \end{cases}$$

［位置ベクトル：$\mathbf{p}$，速度ベクトル：$\mathbf{v}$，加速度ベクトル：$\mathbf{a}$］

シミュレータ作成の方針

- 点のデータを配列に変更して，
  複数の点の運動をシミュレーションする
- 右クリックで点電荷を配置できるようにする
- 加速度を点電荷からの引力・斥力から計算する

加速度：$\mathbf{a} = \dfrac{\mathbf{F}}{m_e}$　(m_eは電子の質量)

力：$\mathbf{F} = q_e \mathbf{E}$　(q_eは電子の電荷)

電界：$\mathbf{E} = \sum_j \dfrac{q_j}{4\pi\varepsilon_0} \dfrac{\mathbf{r}_j}{\|\mathbf{r}_j\|^3}$　($\mathbf{r}_j$は点電荷q_jがある点からのベクトル)

- その他，電気力線や等電位線を描いてみる

参考になる URL

- OpenGL プログラミングについて

  http://www.wakayama-u.ac.jp/~tokoi/opengl/libglut.html

- 微分方程式の数値解法について

  http://www.akita-nct.ac.jp/~yamamoto/lecture/2003/5E/lecture_5E/diff_eq/diff_eq.html

- 電磁気学 静電場について

  http://ja.wikibooks.org/wiki/電磁気学_静電場