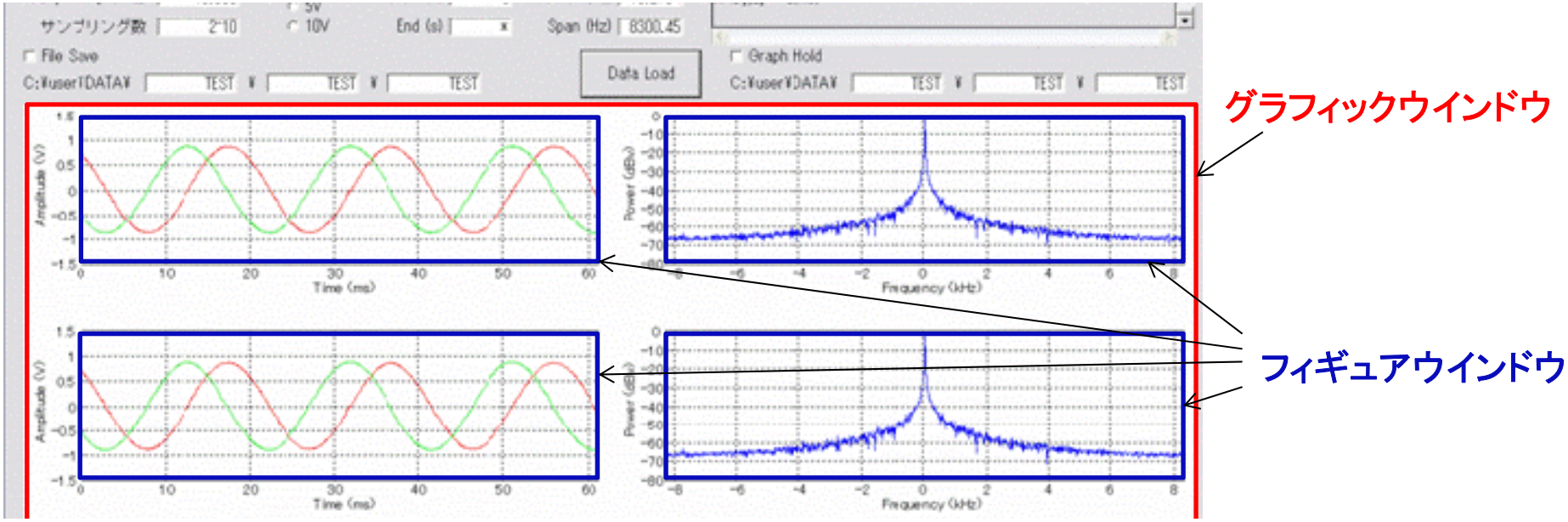
４回目 グラフ作成ライブラリmlibの使い方、図形描画

グラフィックウインドウの説明
Line, Rect, Circle
plot_pen,
アニメーション
総合演習、放物線アニメーション説明
Pause, delay

グラフィックウインドウの設定



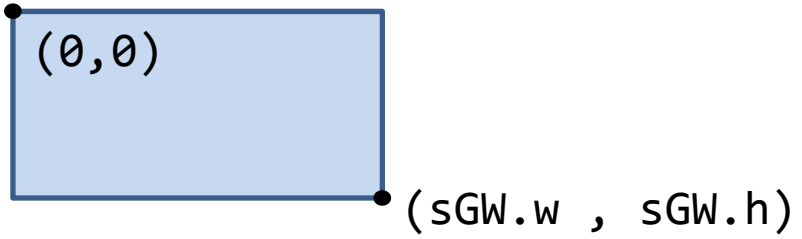
・グラフィックウインドウは図が描画されるウインドウである
・位置、サイズはdef.h内最終行付近のsGW構造体を変更してもよい 。標準では定数 MAINWIN_W と MAINWIN_H を変更すれば自動的に変更されるようになっている。

・Set_figure()関数により、複数のフィギュアウインドウを配置可能である。
・各フィギュアウインドウの左、右、上、下マージンはdef.h 190行目付近 MERGIN_L, MERGIN_R, MERGIN_T, MERGIN_Bで変更できる。
・特にy軸の数値ラベル幅が広いときなど、適宜MERGIN_Lを大きくする必要がある。

・Plot_1d(), Plot_2d()関数等でグラフを作成することができる。

グラフィックウインドウへの図形描画

グラフィックウインドウは独立した座標系
を持ち、ピクセル単位で座標を指定する。
左上(0,0)、右下(sGW.w , sGW.h)




```
void Line(int x1,int y1,int x2,int y2)
```

グラフィックウインドウに直線を引く
x1,y1は始点の座標、x2,y2は終点の座標

```
void Rect(int x1,int y1,int x2,int y2,int bfflug)
```

グラフィックウインドウに四角形を描く
x1,y1は左上の座標、x2,y2は右下の座標

bfflug
- 0 : 枠のみ。枠内は描画しない。
- 1 : 枠と枠内を塗りつぶす
- それ以外 : 枠と枠内を白で塗りつぶす (消去)

```
void Circle(int x1,int y1,int x2,int y2,int bfflug)
```

グラフィックウインドウに楕円を描く。
指定した座標の四角形に接する楕円を描く。
Bfflugの意味はRect関数と同じ

グラフィックウインドウの描画色指定

```
void Plot_pen(int pf, int pw, int pc)
```

グラフィックウインドウに描画する色を指定する。

図形描画だけでなく、グラフのプロット関数の線色、線種指定にも使う

pf :
0 - 実線
1 - 破線
2 - 点線
3 - １点鎖線
4 - ２点鎖線
5 - 描画しない

pw:
ペンの太さ。
ピクセル単位
で指定。
0は1ピクセル

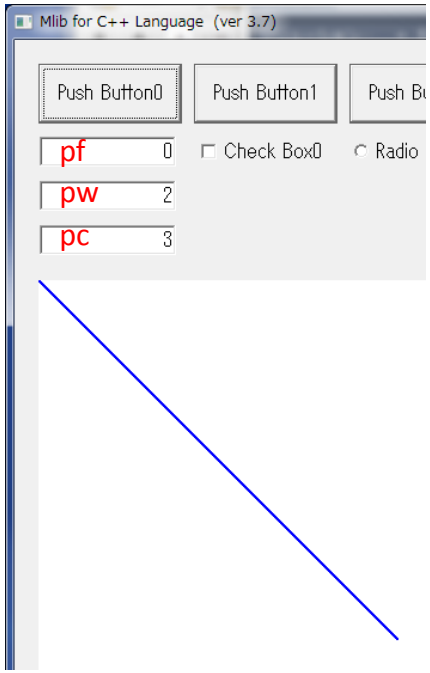
pc:
0-黒
1-赤
2-緑
3-青
4-黄色
5-マゼンダ（水色）
6-シアン（ピンク）
7-白

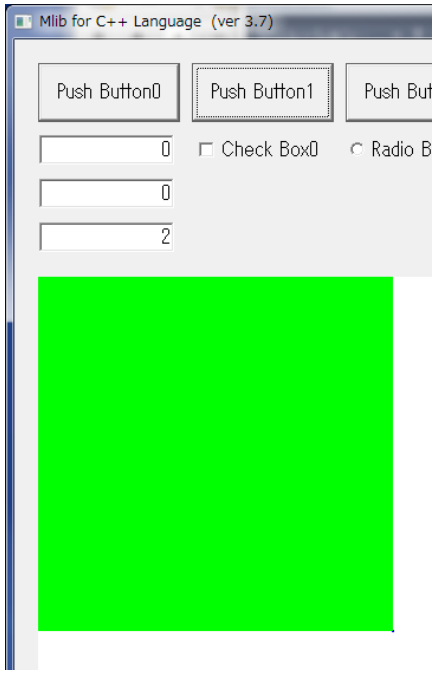
pcに8以上を指定していれば、Plot()関数等のグラフ描画関数を呼び出す毎に自動的に色を0～6まで巡回させる。

色を指定するときは、LINE関数、RECT関数等より前にこの関数で指定する。
1回指定すれば、その色、線種の状態は保持される。

演習6

６－１．エディットボックス0,1,2に入力した整数値をそれぞれ、Plot_pen()関数の３つの引数に対応させて線種の指定を行い、プッシュボタン0,1に対応して、それぞれ、直線、四角を描画するプログラムを作成せよ。尚、プッシュボタン2は四角の範囲を白で塗りつぶし、図の消去を行う



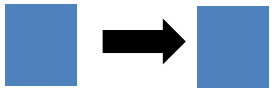


エディットボックスに様々なパラメータを入れ、図を確認してみよう。

```
void main (int Number){
   int a,b,c;
   a=Get_int(0);
   b=Get_int(1);
   c=Get_int(2);

  Plot_penで線種の指定

   if (Number==0){
     直線の描画

   } else if (Number==1) {
     長方形を塗りつぶして描画

   } else{
      Plot_pen(0,1,7);
     長方形を塗りつぶして描画
   }
}
```

物体の移動アニメーション

四角を移動させるには、for文で座標を変えながら表示すればよい？



```
for(i=0;i>100;i++){
   Plot_pen(0,1,2);
   Rect(i,0,i+50,50,1);
}
```

動いた分だけ消さないと移動したことにならない

```
for(i=0;i>100;i++){
   Plot_pen(0,1,2);
   Rect(i,0,i+50,50,1);

   Plot_pen(0,1,7); 白色で描画
   Rect(i,0,i+50,50,1);
}
```

白色で描画することで、書いた■を消す

実行しても何も表示されない。

Line,Rect,Circle関数では、メモリの仮想領域に描画するだけで、画面への更新は行わない

```
for(i=0;i>100;i++){
   Plot_pen(0,1,2);
   Rect(i,0,i+50,50,1);

   Plot_pen(0,1,7);
   Rect(i,0,i+50,50,1);
   UpdateWindow(hWnd);
}
                    画面更新
```

この関数呼び出しで画面が更新される。アニメーションでは描画繰り返し毎に呼び出す必要あり

実行しても書いて、消した後に画面更新しているので、白い■しか表示されない（何も表示されない）

UpdateWindow(hWnd)	強制的に画面の再描画を行う。

物体の移動アニメーション

```
for(i=0;i>100;i++){
    Plot_pen(0,1,2);
    Rect(i,0,i+50,50,1);
    UpdateWindow(hWnd);
    Plot_pen(0,1,7);
    Rect(i,0,i+50,50,1);
}
```

一瞬■が見える

速度が速すぎて動いているように見えない。（画面表示のリフレッシュレート60Hzより早く描画しても無駄）

```
for(i=0;i>100;i++){
    Plot_pen(0,1,2);
    Rect(i,0,i+50,50,1);
    UpdateWindow(hWnd);
    Delay(10);     10ms待つ
    Plot_pen(0,1,7);
    Rect(i,0,i+50,50,1);
}
```

時間稼ぎすることで、動いて見えるが、消えてしまう

アニメーション部の繰り返しの最後が、消去で終わるので、描画した図が、残らない

`void Delay(int msec)` 待ち時間をmsecミリ秒で指定

```
for(i=0;i>100;i++){
    Plot_pen(0,1,7);
    Rect(i-1,0,i+50-1,50,1);
    Plot_pen(0,1,2);
    Rect(i,0,i+50,50,1);
    UpdateWindow(hWnd);
    Delay(10);
}
```

先に、ひとつ前の座標で■を消す。その後、現在の座標に描画

移動の基本

1. 過去の位置の物体を消す
2. 新たな座標の物体を描画
3. 画面の更新
4. 繰り返し時間調整

リアルタイムキー入力

```
Short  GetAsyncKeyState( int key )
```

win32APIの関数でkey に対応する数値（仮想文字コード）のキーが押されているかどうかをリアルタイムに判定する。押されてなければ0、押されていれば負の値を返す。キーとその数値は定数定義されている。

定義定数	押されたキー
VK_BACK	BackSpace
VK_TAB	Tab
VK_RETURN	Enter(Return)
VK_SHIFT	Shift(左右とも)
VK_ESCAPE	Esc

定義定数	押されたキー
VK_SPACE	Space
VK_LEFT	←
VK_UP	↑
VK_RIGHT	→
VK_DOWN	↓

アルファベットは「A」なら’A’のようにシングルコーテーションを使って指定

```
short a=0;

While(a<0){
  a=GetAsyncKeyState( VK_LEFT );
}
Printf(“今、左キーが押されました¥n”);
```

実行しても何も起こらないが、
左向きの矢印キーが押されると、
「今、左キーが押されました」と
表示される。

演習7

7－1．矢印キーの左、右、下、上キーに対応して、円を動かすプログラムを作成せよ。
なお、ESCキーで終了するものとする。

```
void main (int Number){
  short pl,pr,pu,pd;
  int Nx=sGW.w/2,Ny=sGW.h/2;
  int x=0,y=0,r=10;
  while(1) {
    pl=GetAsyncKeyState( 左キー );
    pr=GetAsyncKeyState( 右キー );
    pu=GetAsyncKeyState( 上キー );
    pd=GetAsyncKeyState( 下キー );

    グラフィックウインドウ中心を原点(x,y)=(0,0)
    に設定、この位置に描かれた円を消去する

    if (pl<0) x 座標を減らす(左に移動)
    if (pr<0) x 座標を増やす(右に移動)
    if (pu<0) y 座標を減らす(上に移動)
    if (pd<0) y 座標を増やす(下に移動)

     半径を考慮して、新しい位置に円を描画する

    UpdateWindow(hWnd);
  }
}
```

sGW.w, sGW.h はグラフィックウインドウの幅と高さ

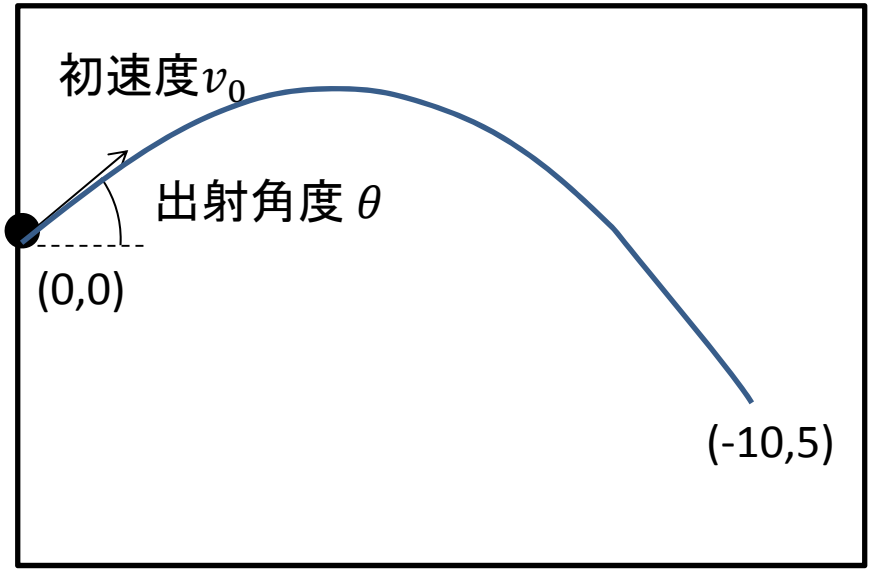
While()文は()内が0になるまで繰り返すので、例を無限ループ

仮想キーコードを指定

球を動かすには
1．過去の座標での円を消去
2．新しい座標を計算
3．その座標に円を描く
の流れ

演習8

A－1．ボールを出射角度 θ 、初速度v_0で投げ上げた時のボールの放物運動のアニメーションを作成せよ。
尚、θ、v_0はそれぞれ、エディットボタン0,1から入力する。
座標系は球の初期位置を原点とする。また、重力加速度は$9.8\mathrm{m}/s^2$とする。



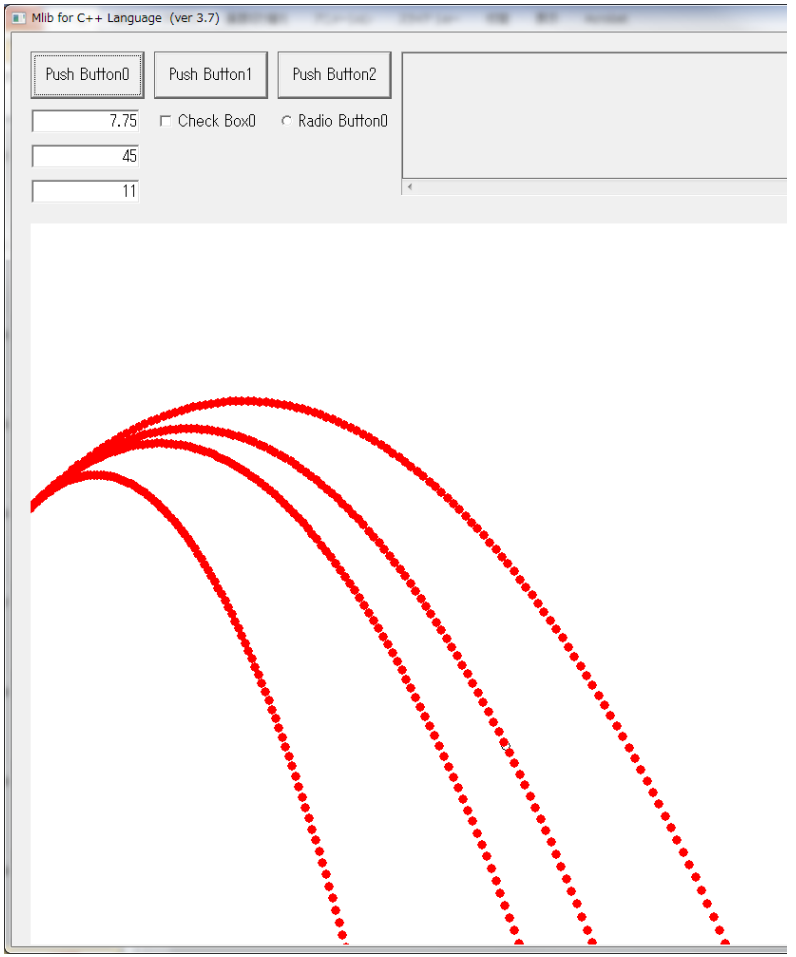


Fig2clipboad() 関数を呼び出すと、グラフィックウインドウ全体の画像をクリップボードにコピーできる。

アルゴリズムのヒント

1. ボールの運動方程式を立てる。

$$ma = mg$$

2. ボールの位置を時間の関数で表す

$$x(t) = (v_0 \cos\theta)t \qquad z(t) = (v_0 \sin\theta)t - \frac{1}{2}gt^2$$

3. ボールの座標系とグラフィックウインドウの座標系の変換式を考える

$$X = f(x) \qquad Z = g(z)$$

4. アニメーションの手法を考える
   例）ある時間間隔 dt 毎のボールの軌跡を描く
   for文等で以下を繰り返し
   - 時間 t のボールの座標を計算
   - 実座標をグラフィックウインドウ座標に変換
   - ボールを描画（Circle 関数）
   - 時間を更新(t = t + dt)