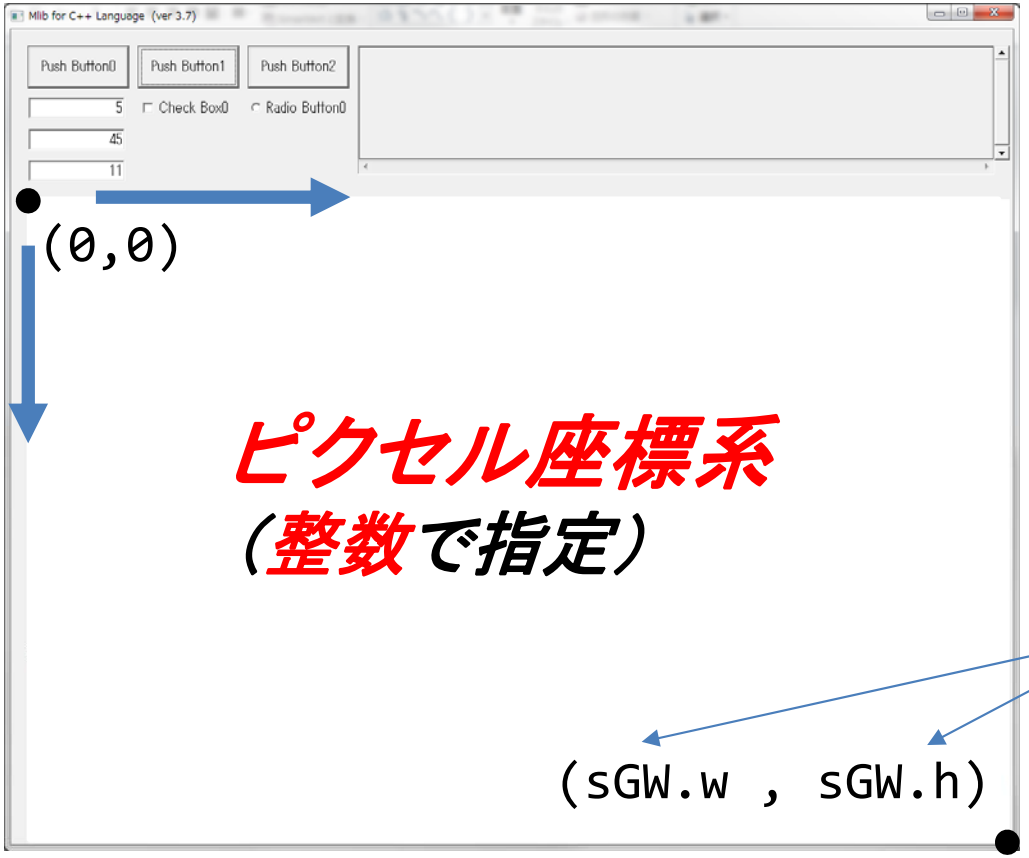
５回目　図形描画とアニメーション

Line(), Rect(), Circle()
アニメーション
座標変換
リアルタイムキー入力

グラフィックウインドウの座標

Plot関数等はグラフィックウインドウ内にフィギュアウインドウを作って、
独立な任意の座標を設定している

グラフィックウインドウ内には、グラフ以外に図形の描画が可能
図形の描画はピクセル単位で指定する
グラフィックウインドウの左上を原点とし、上下は下に増加することに注意



グラフィックウインドウは独立した座標系を持つ

ピクセル単位で座標を指定

左上 (0,0)
右下 (sGW.w , sGW.h)

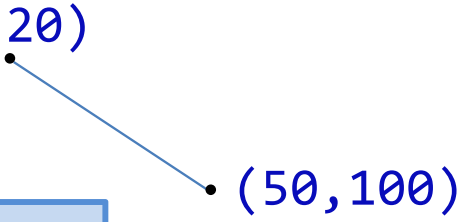
main関数内でこの定数を参照可能

グラフィックウインドウへの図形描画

図形描画関数の座標の指定はすべて整数型

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

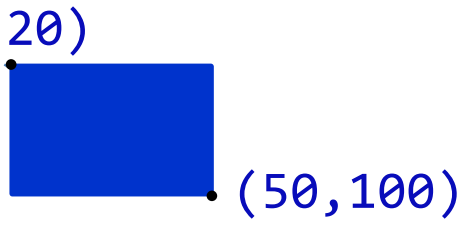
例） **Line(10,20,50,100);**



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)
```

例） **Rect(10,20,50,100,1);**

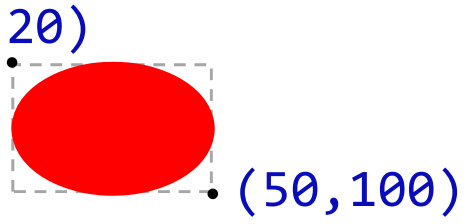


グラフィックウインドウに四角形を描く
x1,y1は左上の座標、x2,y2は右下の座標

bfflug
- 0 : 枠のみ。枠内は描画しない。
- 1 : 枠と枠内をPlot_Penで指定された色で塗りつぶす
- 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)
```

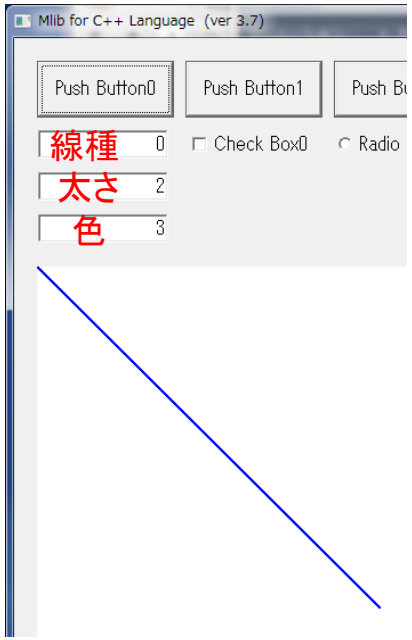
例） **Circle(10,20,50,100,1);**

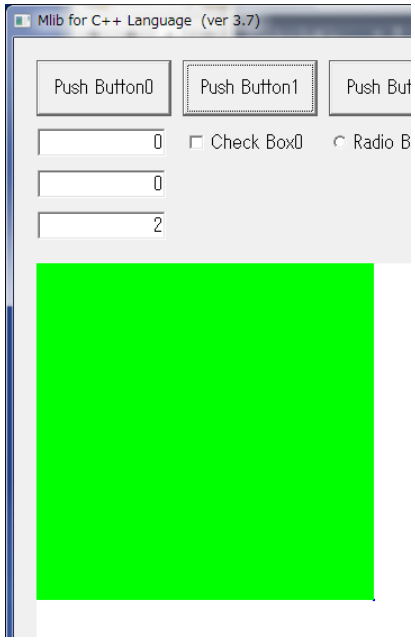


グラフィックウインドウに楕円を描く。
指定した座標の四角形に接する楕円を描く。
Bfflugの意味はRect関数と同じ

例題５－１

５－１．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





線種：
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

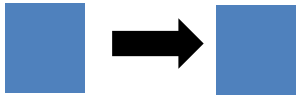
太さ:
ペンの太さ。
ピクセル単位で指定。
0は1ピクセル

色:
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

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

   Plot_pen(a,b,c);

   if (Number==0){
      Line(0,0,300,300);

   } else if (Number==1){
      Rect(0,0,300,300,1);

   } else {
      Plot_pen(0,1,7);
      Rect(0,0,300,300,1);
   }
}
```

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

```
UpdateWindow(hWnd);
```

強制的にメモリの仮想領域の再描画を行う。

```
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）

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(ii,0,ii+50,50,1); //消去
    Plot_pen(0,1,2);
    Rect(i,0,i+50,50,1);   //描画
    UpdateWindow(hWnd);
    Delay(10);
    ii=i;      //前の座標を更新
}
```

過去の座標iiを導入

先に、ひとつ前の座標iiで■を消す。

その後、現在の座標に描画

移動の基本（以下を繰り返す）

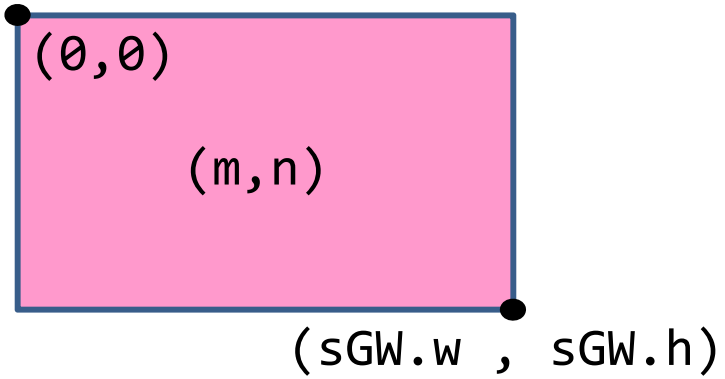
1. 過去の座標にある物体を消す
2. 計算した新たな座標に物体を描画
3. 画面の更新
4. 過去の座標に現在位置を代入

実座標とピクセル座標

ピクセル座標 (m, n)

・左上が原点で右下方向に増える

・座標の値は整数となっている。



放物運動、ボールの跳ね返り等の物理現象のシミュレーションでは、物体の大きさや、速度、実際の座標系で考える方が直感的に分かり易い。

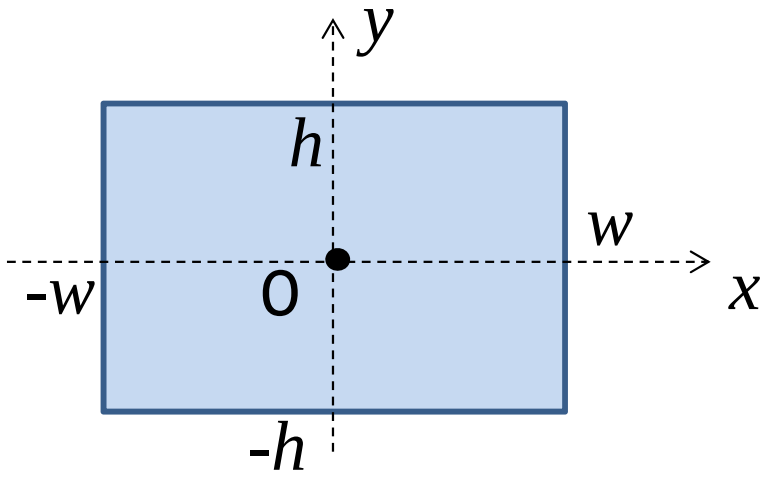
実座標 (x , y)

・物理現象を考えるうえでの実際の座標

・原点はどこでもよい（基本は真ん中）

・右上に向かって増加

・値は実数



（幅2w、高さ2h、原点中央の場合）

実座標とピクセル座標の変換

メインプログラム内での物理現象の数値シミュレーション

→ 実座標を使えば便利

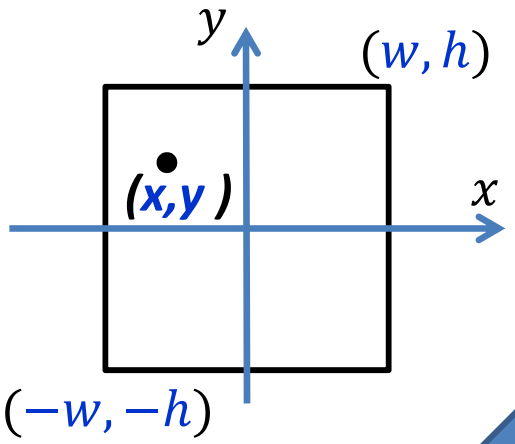
座標変換

グラフィックウインドウへの物体の描画、消去

→ ピクセル座標を使えば便利

中心原点，幅 $2w$，高さ $2h$ の長方形空間をピクセル座標に射影

実座標



・物体の移動、衝突等は**実座標**で考える。

$$\mathtt{m} = \frac{\mathtt{sGW.w}}{2}\left(1 + \frac{x}{w}\right)$$

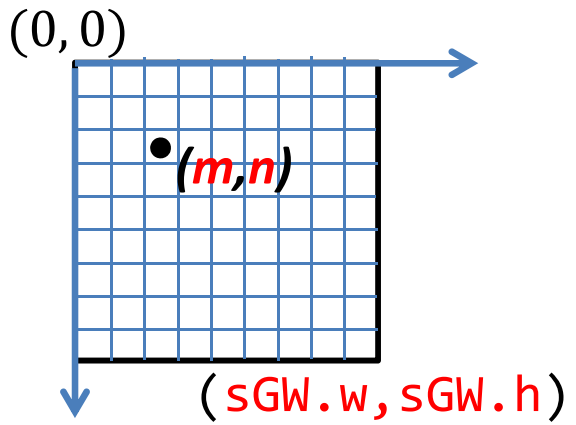
$$\mathtt{n} = \frac{\mathtt{sGW.h}}{2}\left(1 - \frac{y}{h}\right)$$

線形変換による座標変換

$$x = \frac{w(2\mathtt{m} - \mathtt{sGW.w})}{\mathtt{sGW.w}}$$

$$y = \frac{h(\mathtt{sGW.h} - 2\mathtt{n})}{\mathtt{sGW.h}}$$

ピクセル座標



画面表示を行う際に実座標を**ピクセル座標**に変換

実座標での移動プログラム例５－２

```
void main(int Number){
int i,m,n,mm,nn;
double x=0,y=0,w=100,h=100;

mm=0;nn=0;
for(i=0;i<500;i++){

   Plot_pen(0,1,7); //白色に設定（バックと同じ）
   Rect(mm,nn,mm+50,nn+50,1); //過去の位置の図を消去

   x=x+0.1;  //実座標を更新

   m=sGW.w/2*(1+x/w);  //実座標をピクセル座標に変換
   n=sGW.h/2*(1-y/h);  //実座標をピクセル座標に変換

   Plot_pen(0,1,2); //緑色に指定
   Rect(m,n,m+50,n+50,1);  //新しい位置に図を描画

   UpdateWindow(hWnd);  //画面更新

   mm=m; //新しいピクセル座標を過去のピクセル座標にする
   nn=n; //新しいピクセル座標を過去のピクセル座標にする
 }
}
```

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５－３

５－３．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完成せよ。

```
void main(int Number){
 short pl,pr,pu,pd; // キー入力判定用
 int m,n,mm=0,nn=0,r=10;  //物体の現在、過去の位置のピクセル座標保存用
 double x=0,y=0,w=100,h=100;   //物体の現在位置の実座標、実座標の範囲
 while(1) {          //While()文は()内が0になるまで繰り返すので、例では無限ループ
   pl=GetAsyncKeyState( 左キー );
   pr=GetAsyncKeyState( 右キー );
   pu=GetAsyncKeyState( 上キー );
   pd=GetAsyncKeyState( 下キー );

   Plot_pen(0,2,7);                          //白色に設定（バックと同じ）
   過去の位置に描かれた円を消去する。半径はr
    x 座標を減らす（左に移動）
    x 座標を増やす（右に移動）
    y 座標を減らす（上に移動）
    y 座標を増やす（下に移動）
   m=sGW.w/2*(x/w+1);                        //実座標のx座標をピクセル座標に変換
   n=sGW.h/2*(1-y/h);                        //実座標のy座標をピクセル座標に変換

   Plot_pen(0,2,3);                          //緑色に指定
   新しい位置に描かれた円を描画する。半径はr
   UpdateWindow(hWnd);                       //画面更新
   mm=m; nn=n;                               //新しいピクセル座標を過去のピクセル座標にする
 }}
```

仮想キーコード入力判定

それぞれのキーが押されていれば座標を更新