回転する矩形と点の当たり判定

回転する矩形（四角形）と点の当たり判定について説明します。まず、作業しやすくするために、矩形を表す構造体を作成します。次に、水平に置かれた矩形の当たり判定を考えてから、傾いた矩形を考えるとわかりやすいです。

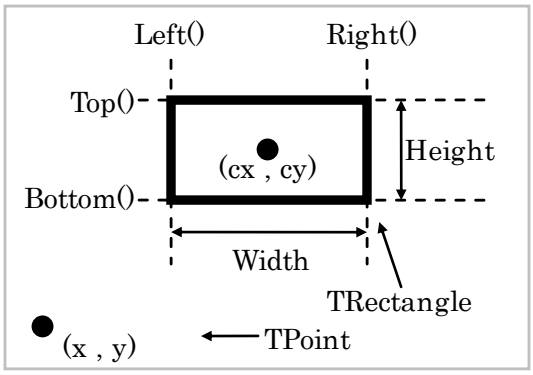
- 矩形構造体の定義（TRectangle）

後の傾いた矩形の当たり判定を作りやすくするために、矩形の構造体 TRectangle を宣言します。

```
struct TRectangle
{
    double  cx, cy;          //中心座標
    int     Width, Height;   //回転前の横幅,縦幅
    double  r;               //回転角（ラジアン）

    //回転前の座標を求める
    double  Left()  {return cx - Width/2;}  //左
    double  Top()   {return cy - Height/2;} //上
    double  Right() {return cx + Width/2;}  //右
    double  Bottom(){return cy + Height/2;} //下
};
```

矩形は中心座標(cx, cy)と横幅(Width),縦幅(Height)で表します。また、各々の辺の座標はメソッドで求められるようにしています。



- 点の構造体（TPoint）[1]

C++Builder 標準の構造体 TPoint を使います。メンバ変数に x, y がある単純な構造体です。[2]

---

[1] TPoint を扱うには<vcl.h>または<Windows.hpp>がインクルードされている必要があります。

[2] Windows 標準の POINT 構造体でも同様です

- 水平に置かれた矩形と点の当たり判定

いわゆる普通の「四角形と点の当たり判定」です。既に熟知している方は読み飛ばしてもらって構いません。

✓ 与えられる構造体は

1. 矩形の TRectangle
2. 点の TPoint

の２つです。

✓ 当たり判定は点(x, y)を中心に考えて、

x が Left と Right の間にあり、y が Top と Bottom の間にある

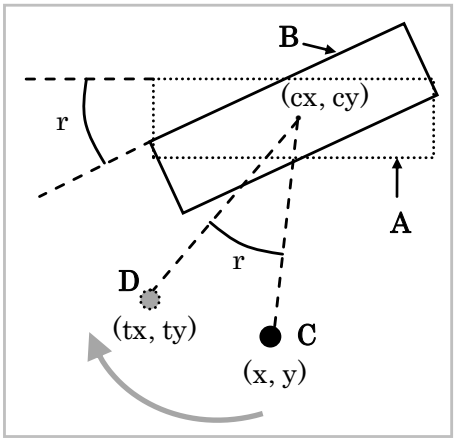
という条件を満たせば当たっている事になります。

◎ サンプル関数

引数に TRectangle と TPoint を入れ、それらが当たっている時は１を返し、当たっていない時は０を返します。

```
int HitRect(TRectangle rect, TPoint pt)
{
    if(rect.Left() <= pt.x && pt.x <= rect.Right() &&
       rect.Top() <= pt.y && pt.y <= rect.Bottom()){
        return 1;
    }
    return 0;
}
```

● 傾いた矩形と点の当たり判定



1 ドットしか当たり判定がないマウスポインタが、回転している壁と当たっているかどうか調べるなどという用途に使えます。

右図を見てください。回転前の矩形を A、角度 r だけ回転させた矩形を B とします。この矩形 B と点 C の当たり判定を考えます。これは、**点 C を反対方向に角度 r だけ回転させた架空の点 D と矩形 A との当たり判定**を行うことと同じです。

手順としては

1. 架空の点 D の座標を求める
2. 水平の矩形と点の当たり判定を行う

で判定をすることができます。

1. <u>架空の点 D を求める</u>

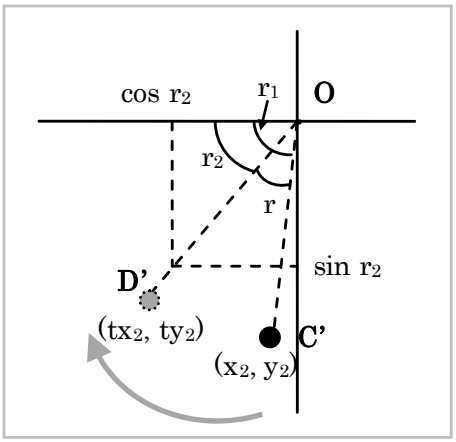
矩形の中心座標を(cx, cy)、点 C を(x, y)、架空の点 D を(tx, ty)とします。

点 C と矩形の中心との距離 l は

$$l = \sqrt{(x-cx)^2 + (y-cy)^2} \qquad \cdots(1)$$

で求まります。

矩形の中心座標を原点として見た場合の相対的な点 C を C'(x_2, y_2)、同様に点 D を D'(tx_2, ty_2)とします。



$$x_2 = x - cx$$
$$y_2 = y - cy$$
$$tx_2 = tx - cx$$
$$ty_2 = ty - cy \qquad \cdots(2)$$

横軸に対する C'の角度を r_1、D'の角度を r_2 とします。ここで r_2 を求めます。

$$r_1 = \tan^{-1}\frac{y_2}{x_2} \qquad \cdots(3)$$
$$r_2 = r_1 - r$$

D'と原点を結ぶ直線上で、原点から長さ 1 だけ離れた点の座標は (cos r2, sin r2)です。これに長さ l を掛けると D'の座標が求まります。

$$tx_2 = l \times \cos r_2$$
$$ty_2 = l \times \sin r_2 \qquad \cdots(4)$$

点 D'(tx2, ty2)を点 D(tx,ty)に戻します。

$$tx = tx_2 + cx$$
$$ty = ty_2 + cy$$

ここまで求まれば、点 D と水平な矩形の当たり判定を行えば完成です。

◎ サンプル関数

```
#ifndef HITRECTANGLE_H
#define HITRECTANGLE_H

#include <math.h>
#include <Windows.hpp>
#include "TRectangle.h"
//---------------------------------------------------------------------
int HitRectAngle(TRectangle rect, TPoint pt)
{
    // (1)矩形の中心と点の距離を計算
    double l;
    l = sqrt(pow(pt.x - rect.cx, 2) + pow(pt.y - rect.cy, 2));

    // (2)矩形の中心を原点として見た相対的な点C'の座標
    TPoint pt2;
    pt2.x = pt.x - rect.cx;
    pt2.y = pt.y - rect.cy;

    // (3)点D'と横軸のなす角r2を求める
    double r1, r2;
    if(pt2.x != 0){r1 = atan(pt2.y / pt2.x);}
    else{r1 = M_PI_2;}          //0による除算を回避
    r2 = r1 - rect.r;

    // (4)点D'の座標
    TPoint pt3;
    pt3.x = l * cos(r2);
    pt3.y = l * sin(r2);

    // (5)点Dに戻す
    pt3.x += rect.cx;
    pt3.y += rect.cy;

    // 普通の矩形と点の当たり判定
    if(rect.Left() <= pt3.x && pt3.x <= rect.Right() &&
       rect.Top() <= pt3.y && pt3.y <= rect.Bottom()){
        return 1;   //当たっている
    }
    return 0;       //当たっていない
}
//---------------------------------------------------------------------
#endif
```

まだまだ計算量が多いので、もっとスマートに書ける方法があると思います。これを改良してより良い物を作っていってください。

参考：

http://f55.aaa.livedoor.jp/~mclass/hsplecture/nanamekukei.htm