「　三角関数２　」

東京情報クリエイター工学院

三角関数２

◆三角関数(復習)

三角関数

単位円の円周上の点の　x　座標は$cos\theta$　　$x = cos\theta$

単位円の円周上の点の　y　座標は$sin\theta$　　$y = sin\theta$

原点と単位円の円周上の点を結ぶ直線の傾きが$tan\theta$　　$tan\theta = \frac{y}{x}$

三角関数の相互関係

$sin(90° - \theta) = cos\,\theta$　　$sin(\theta + 90°) = cos\,\theta$

$cos(90° - \theta) = sin\,\theta$　　$cos(\theta + 90°) = -sin\,\theta$

$sin(180° - \theta) = sin\ \theta$　　$cos(180° - \theta) = -cos\,\theta$

$sin(\theta + 180°) = -sin\,\theta$　　$cos(\theta + 180°) = -cos\,\theta$

弧度法の定義

半径１の円において、**半径と同じ長さの弧をもつ扇形**を考えます。この**扇形の中心角を「１ラジアン(rad)」**とします。

弧度法と度数法の関係】

「$180°$ ＝π ラジアン」の関係であり、

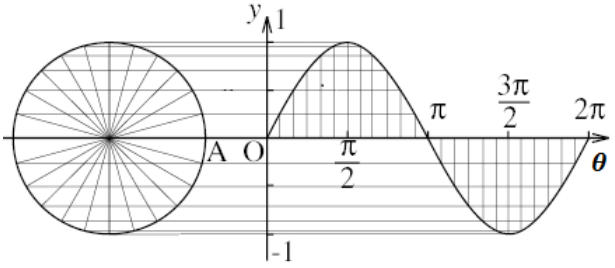
[ラジアンで表した角]＝[度で表した角]× $\frac{\pi}{180}$

[度で表した角]＝[ラジアンで表した角]× $\frac{180}{\pi}$

◆三角関数のグラフ

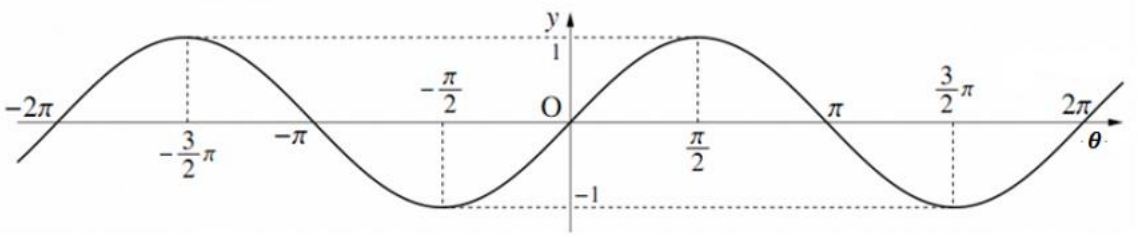
１．$y = sin\theta$ のグラフ

三角関数ということは、関数なわけですから、1 次関数(直線)、2 次関数(放物線)のように、グラフを使って視覚化できます。



左にある単位円の動径をぐるっと 1 回転させたときの θ の変化を横軸に、そしてそのときの y 座標を縦軸に、右のグラフに順次とっていくと、上のような形になります。この曲線を「　**正弦曲線**　」または「　**サインカーブ**　」と呼びます。

下のグラフを見てください。



「　三角関数２　」

東京情報クリエイター工学院

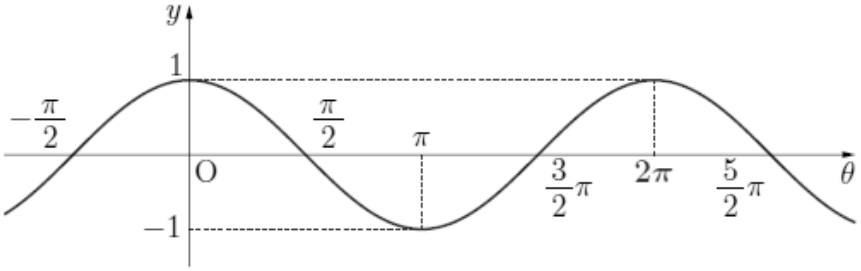
区間 $-2\pi \leqq \theta \leqq 0$ のグラフと区間 $0 \leqq \theta \leqq 2\pi$ のグラフの形は完全に一致していることがわかりますね。これは、動径が１回転することで元の位置に戻ることから当然の性質ですね。

この性質から、$y = \sin\theta$ は、2π の「　　」をもつ「 **周期関数** 」といえます。また、振動の幅は、$1 \leqq y \leqq -1$ ですね。この振動の幅の半分を「　　」といいます。従って、$y = \sin\theta$ の振幅は、「 1 」です。また、このグラフは、原点対象となるので、「奇関数」となります。

2. $\cos$　$\tan$ のグラフ

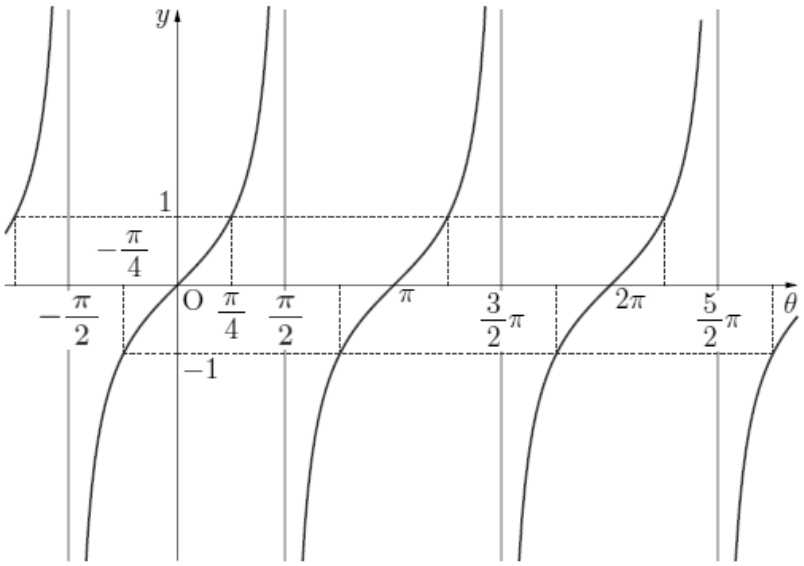
$y = \cos\theta$

値域：$-1 \leqq y \leqq 1$，　周期：2π，　偶関数 (y 軸対称)



$y = \tan\theta$

定義域：$\theta \neq \frac{\pi}{2} + n\pi$ (n：整数)，　周期：π，　奇関数 (原点対称)



◆三角関数のグラフ

三角関数の意味を知るために、以下のプログラムを作成してみましょう。今までのことを参考に、どのように作成すればよいか考えましょう。$\sin$ のグラフから作成しましょう。

０．プログラム作成の準備

(1) プログラムを起動していたら、プログラムを終了する。

(2) [C_学籍番号_氏名]フォルダを開き、[MathCalc]フォルダの中の[MathGraph00]フォルダをコピーし、名前を[MathGraph01]に変更する。

(3) 名前を変更したら、[MathGraph01]フォルダを開き、[MathGraph00.csproj]の名前を[MathGraph01.csproj]に変更する。

(4) [MathCalc.sln]を起動する。

(5) ソリューションエクスプローラのソリューションを右クリック－[追加]－[既存のプロジェクト]

「　三角関数２　」

東京情報クリエイター工学院

(6)「既存のプロジェクトの追加」で、先ほど名前を変更した[MathCalc]フォルダー[MathGraph01]フォルダー[MathGraph01.csproj]をクリックし、[開く]ボタンをクリックする。

(7)ソリューションエクスプローラに[MathGraph01]が追加される。

(8)[MathGraph01]をスタートアップ プロジェクトに設定する。

１．[MathGraph01]を使用し、三角関数(sin)を描くプログラムを作成しなさい。

(1)[MyDrawClass.cs]の初期値を変更し、三角関数の式をプログラムに入れましょう。

```
class MyDrawClass:Draws
{
    Vector2 p0;  //座標
    float x, y, a, b, step;
    //float c;
    //float r; //修正
    float scaleX, scaleY; //倍率

    public MyDrawClass() { }

    public void graph()
    {
        x = -10.0f;  //x の初期値
        a = 1.0f;     //増幅
        b = 1.0f;     //周期
        //c = 0.0f;
        //r = 100.0f;   //半径 コメントにする
        step = 0.1f; //x の増分
        scaleX = 20.0f;   //x の倍率
        scaleY = 100.0f;  //y の倍率

        Clear();

        //グラフの描画
        while (x * scaleX < 200 && y * scaleY < 200)
        {

            y = a * (float)Math.Sin(x * b);  //Sin の式
            p0 = new Vector2(x * scaleX, y * scaleY);
            Dot(p0);
            Render();

            x = x + step;  //x の値を step 増加
        }
    }
}
```

(2)式を変更して、振幅・周期の変化を確認しましょう。

sin の式で、振幅($a = 2.0f$)を変更してみましょう。

変更したら、実行してみましょう。先ほどの sin のグラフとどう変化していますか？

今度は周期を変更してみましょう。($a = 1.0f$ $b = 2.0f$)

２．sin の式を cos に変更してみましょう。

３．sin,cos で円を描くプログラムを作成しなさい。

初期値を変更し、三角関数を使って円を描きましょう。

式・描画の際の While 文の条件を変更する必要がありますね。

「　三角関数２　」

東京情報クリエイター工学院

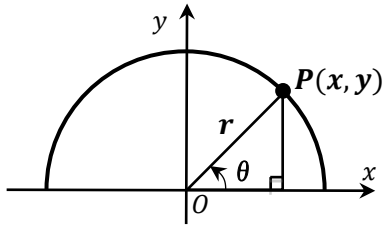
◆任意の点の回転

１．原点で回転した点の座標

前回学習したように、$\boldsymbol{cos}$ は**単位円の円周上の点の** $\boldsymbol{x}$ 座標であり、$\boldsymbol{sin}$ は単位円の円周上の点の $\boldsymbol{y}$ 座標でした。

今度は、視点を変えて、$\boldsymbol{xy}$ 平面上の任意の点 $\boldsymbol{P}$ の x 座標 $\boldsymbol{y}$ 座標を考えてみましょう。

原点を中心に回転半径 $\boldsymbol{r}$ として $\boldsymbol{\theta}$ 回転した点 $\boldsymbol{P'}$ の座標はどうなりますか？

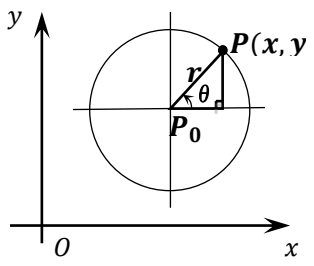


$\boldsymbol{x}$ 座標　　「　　」

$\boldsymbol{y}$ 座標　　「　　」

２．原点の移動

次に、原点を $\boldsymbol{P_0(x_0, y_0)}$ へ移動すると点 $\boldsymbol{P(x, y)}$ の座標はどうなるでしょうか？



原点　$\boldsymbol{(0, 0)}$　→　$\boldsymbol{(x_0, y_0)}$

$\boldsymbol{x}$ 座標　　「　　」

$\boldsymbol{y}$ 座標　　「　　」

【練習問題１】

角度　0°の位置から指定された角度で回転した座標値を求めなさい。

①原点を中心に回転半径 10、角度を 30°で回転した座標値。

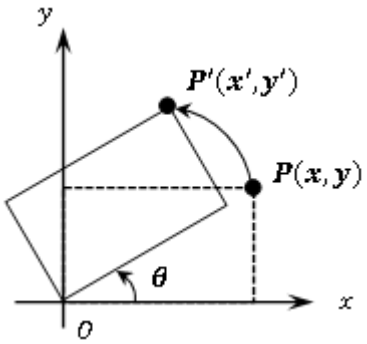
②　P(20, 30)の点を中心に回転半径 6、角度を 60°で回転した座標値。

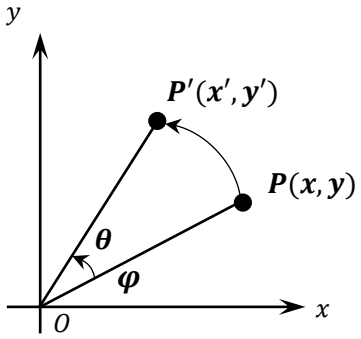
「　三角関数２　」

東京情報クリエイター工学院

３．原点を中心とする任意の回転

下図のような任意点 $\boldsymbol{P}$ を原点中心に $\boldsymbol{\theta}$ 回転することを考えます。





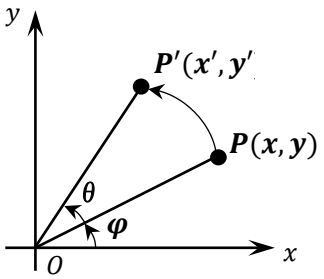
このとき任意点 $\boldsymbol{P(x,\ y)}$ は、中心からの距離を $\boldsymbol{r}$ とすると、

x **座標**　「　　」　　y **座標**　「　　」

移動点 $\boldsymbol{P'(x',\ y')}$ は、

x **座標**　「 $\boldsymbol{x\cos\theta - y\sin\theta}$ 」　　y **座標**　「 $\boldsymbol{x\sin\theta + y\cos\theta}$ 」

となります。何故、このようになるのか考えてみましょう。



原点を中心に回転半径 $\boldsymbol{r}$ 、角度を $\boldsymbol{\varphi}$ の位置にある点 $\boldsymbol{P(x,\ y)}$ は、

x **座標**　「　　」　　y **座標**　「　　」

この点 $\boldsymbol{P(x,\ y)}$ をさらに $\boldsymbol{\theta}$ 回転させた点 $\boldsymbol{P'(x',\ y')}$ は

x **座標**　「　　」

y **座標**　「　　」

ここで、以下の公式を用いて、移動点 $\boldsymbol{P'(x',\ y')}$ を求めます。

加法定理

$$\boldsymbol{\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta}$$
$$\boldsymbol{\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta}$$

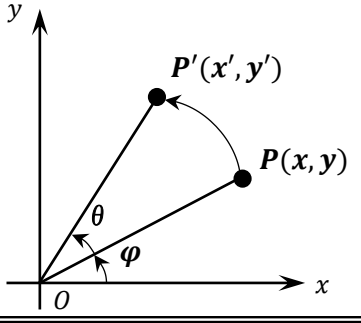
$$\boldsymbol{x' = r\cos(\varphi+\theta) = r(\cos\varphi\cos\theta - \sin\varphi\sin\theta) = r\cos\varphi\cos\theta - r\sin\varphi\sin\theta}$$
$$\boldsymbol{= x\cos\theta - y\sin\theta}$$
$$\boldsymbol{y' = r\sin(\varphi+\theta) = r(\sin\varphi\cos\theta + \cos\varphi\sin\theta) = r\sin\varphi\cos\theta + r\cos\varphi\sin\theta}$$
$$\boldsymbol{= x\sin\theta + y\cos\theta}$$

原点を中心とする任意点の回転

原点を中心に回転半径 r 、角度を $\boldsymbol{\varphi}$ の位置にある点 $\boldsymbol{P(x,\ y)}$ を $\boldsymbol{\theta}$ 回転させた点 $\boldsymbol{P'(x',\ y')}$

x **座標**　「　　」

y **座標**　「　　」



「　三角関数２　」

東京情報クリエイター工学院

[MathGraph01　円のプログラム例]

```
class MyDrawClass:Draws
{
   Vector2 p0;  //座標
    float x, y, a, b, step;
    //float c;
    float r; //修正　コメント解除
    float scaleX, scaleY; //倍率
    float th;             //追加　角度

    public MyDrawClass() { }

    public void graph()
    {
        x = 0.0f;  //x の初期値
        y = 0.0f;  //追加　y の初期値
        a = 0.0f;     //中心の x 座標
        b = 0.0f;     //中心の y 座標
        //c = 0.0f;
        r = 100.0f;   //修正 コメント解除　半径
        th = 0.0f;   //追加
        step = 2.0f * (float)Math.PI / 50.0f;  //修正　th の増分
        scaleX = 1.0f;   //x の倍率
        scaleY = 1.0f;  //y の倍率

        Clear();

        //グラフの描画
        while ((0<=th)&&(th<=2.0f*(float)Math.PI))
        {
            x = r * (float)Math.Cos(th) + a;  //x 座標
            y = r * (float)Math.Sin(th) + b;  //y 座標

            p0 = new Vector2(x * scaleX, y * scaleY);
            Dot(p0);
            Render();

            th = th + step;  //th の値を step 増加
        }
    }
}
```

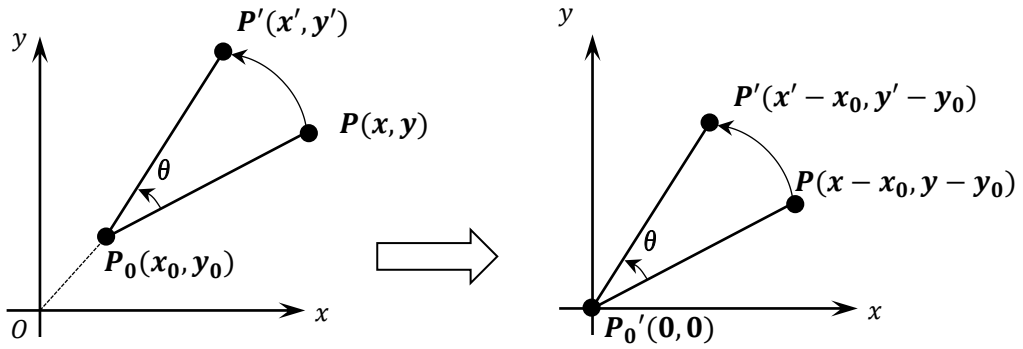
「　三角関数２　」

東京情報クリエイター工学院

４．任意の回転中心から任意点を回転

最後に、任意の回転中心から任意点を回転することを考えます。

任意の点 $\boldsymbol{P_0}$ を回転中心として、下図のような回転後の点 $\boldsymbol{P'}$ を求めるために回転中心を原点に移動すると $\boldsymbol{P_0(x_0\ ,\ y_0)}$ の分だけ平行移動することになります。



後は、先ほど求めた原点を中心とした回転の式に代入することで、求めることができます。

$$\boldsymbol{x' - x_0 = (x - x_0)cos\,\theta - (y - y_0)\,sin\,\theta}$$

x 座標　「 $\boldsymbol{x' = (x - x_0)cos\,\theta - (y - y_0)sin\,\theta + x_0}$ 」

$$\boldsymbol{y' - y_0 = (x - x_0)sin\,\theta + (y - y_0)\,cos\,\theta}$$

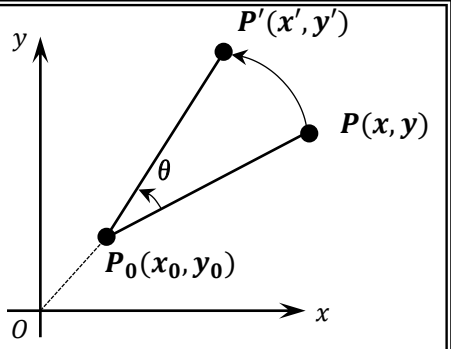
y 座標　「 $\boldsymbol{y' = (x - x_0)sin\,\theta + (y - y_0)cos\,\theta + y_0}$ 」

任意の回転中心による任意点の回転

任意の点 $\boldsymbol{P_0(x_0\ ,\ y_0)}$ を中心に $\boldsymbol{\theta}$° 回転させた点 $\boldsymbol{P'(x'\ ,\ y')}$

x 座標

y 座標



【練習問題２】

次の点を回転した後の座標点を求めなさい。

①原点中心に点 $\boldsymbol{P(10, 20)}$ を $\boldsymbol{30}$°回転する。

②点 $\boldsymbol{P_0(15, 25)}$ を中心に、点 $\boldsymbol{P(35, 15)}$ を $\boldsymbol{45}$°回転する。

「　三角関数２　」

東京情報クリエイター工学院

◆回転のプログラム

１．マウス入力された座標点を原点を中心に回転させるプログラムを作成しなさい。

(1)準備

①ソリューション[MathCalc]の中にテンプレートを使用して[Math02Ex03]を作成する。

②はじめは回転中心を原点、入力点を回転させる座標、回転角度はテキストボックスで入力します。
（マウス入力：１つ）

③テキストボックスを１つ追加

[ツールボックス]を表示ー[label]をクリックしてドラッグ
[label1]のプロパティの[Text]を「回転角度」に変更する

[label1]の下に、[ツールボックス]を表示ー[TextBox]をクリックしてドラッグ
[textBox1]のプロパティーー[Modifiers]を[Public]にする

(2)プログラムの入力

【MyDrawClass.cs】

```
using System.Windows.Forms;
namespace Math02Ex03
{
    public class MyDrawClass : Draws, IDraws
    {
        //フィールド
        InputState input;
        Vector2 p0, p1, pc, pd;
            //修正  p0:回転中心    p1:回転前の座標  pc：任意の回転中心  pd:回転後の座標
        float angle;        //追加
        public Form1 form;  //追加
        String ang;         //追加

        //コンストラクタ
        public MyDrawClass()
        {
            input = new InputState();
            input.MouseOn();
            p0 = Vector2.Zero;   //追加
        }
        //メソッド
        public void InputData()
        {
            Init();
            Clear();
            List<PointF> p = new List<PointF>();
            p = input.GetPoint(1, "座標点入力");  //コメント修正
            if (p.Count == 0) { return; }
            p1 = new Vector2(p[0].X, p[0].Y);    //修正

            //以降  追加
            //原点を中心とする回転
            ang = form.textBox1.Text;   //textBox1のプロパティーデザインーModifiersをPublicにする
            angle = Convert.ToSingle(ang);  //角度の取得
            angle = angle * (float)Math.PI / 180.0f;  //度をラジアンに変換

            float cos = (float)Math.Cos(angle);
            float sin = (float)Math.Sin(angle);

            pd.X =                          //回転後のX座標
            pd.Y =                          //回転後のY座標

            Draw();
            Render();
        }
```

「　三角関数２　」

東京情報クリエイター工学院

```
        public void Update() { }
        public void Draw()
        {
            SetColor(Color.Blue);
            Dot(p1,5);
            Text(p1, "元の座標" + p1);
            Line(p0, p1);

            SetColor(Color.Red);
            Dot(pd,5);
            Text(pd, "変換後の座標" + pd);
            Line(p0, pd);
        }
    }
```

【Form1.cs】追加：[Form1.cs]右クリック－[コードの表示]

```
    public Form1()
    {
            :
        //追加
        draw.form = this;
    }
```

２．１のプログラムを参考にして、最初にマウス入力された点を回転中心とし、次にマウス入力された点を回転させるプログラムに修正しなさい。

「　三角関数２　」

東京情報クリエイター工学院

◆発展問題

１．ゲームのキャラクターが空中の標的に矢を射る場合を想定します。標的は地面から 400 ピクセルの高さの場所にあり、キャラクターは標的から水平距離 100 ピクセル離れた場所にいます。矢が直線経路に沿って飛んでいくとすると、空中の標的を射るにはキャラクターはどの角度で狙いを定めればよいでしょうか。なお、角度は三角関数表から求めた最も近い角度とします。

２．次の数を小さいものから順に並べなさい。ただし、三角関数表を使用しないこと。

$\cos 15^\circ \quad \sin 82^\circ \quad \sin 142^\circ \quad \cos 204^\circ$

３．$0 \leqq \theta < 2\pi$ のとき、 $2\sin\theta - 1 = 0$ を解きなさい。

４．$0 \leqq \theta < 2\pi$ のとき、 $\sqrt{3} - 2\cos\theta = 0$ を解きなさい。

５．$0 \leqq \theta < 2\pi$ のとき、 $2\sin^2\theta - \sin\theta = 0$ を解きなさい。

「　三角関数２　」

東京情報クリエイター工学院

【MyDrawClass.cs プログラム例】

```
pd.X = p1.X * cos - p1.Y * sin;     //回転後のX座標
pd.Y = p1.X * sin + p1.Y * cos;     //回転後のY座標
```

【MyDrawClass.cs】修正

```
public void InputData()
{
    Init();
    Clear();
    List<PointF> p = new List<PointF>();
    p = input.GetPoint(2, "回転中心と座標点入力");  //修正
    if (p.Count == 0) { return; }
    pc = new Vector2(p[0].X, p[0].Y);      //修正  pcに回転中心座標セット
    p1 = new Vector2(p[1].X, p[1].Y);      //追加  p1に座標セット

    //修正  pcを中心とする回転
    ang = form.textBox1.Text;   //textBox1のプロパティーデザインーModifiersをPublicにする
    angle = Convert.ToSingle(ang);  //角度の取得
    angle = angle * (float)Math.PI / 180.0f;  //度をラジアンに変換

    float cos = (float)Math.Cos(angle);
    float sin = (float)Math.Sin(angle);

    pd.X = (p1.X - pc.X) * cos - (p1.Y - pc.Y) * sin + pc.X;     //修正  回転後のX座標
    pd.Y = (p1.X - pc.X) * sin + (p1.Y - pc.Y) * cos + pc.Y;     //修正  回転後のY座標

    Draw();
    Render();
}
public void Update() { }
public void Draw()
{
    Dot(pc, 5);    //追加  回転中心
    Text(pc, "回転中心" + pc);   //追加

    SetColor(Color.Blue);
    Dot(p1, 5);
    Text(p1, "元の座標" + p1);
    Line(pc, p1);   //修正

    SetColor(Color.Red);
    Dot(pd, 5);
    Text(pd, "変換後の座標" + pd);
    Line(pc, pd);   //修正

}
```