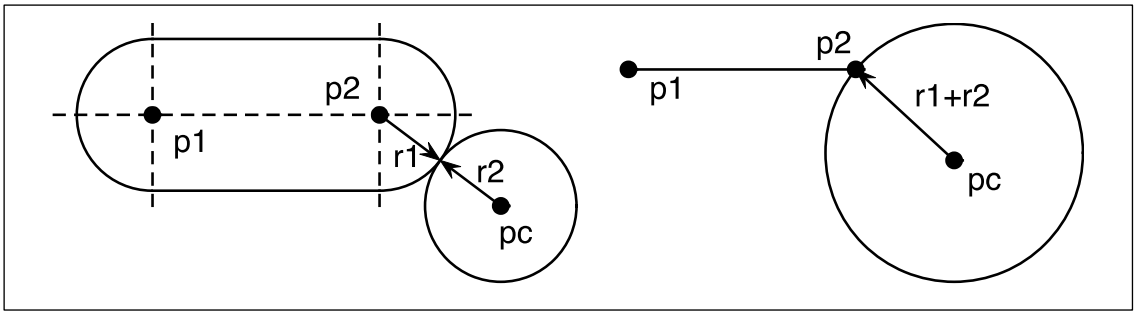
「　ベクトルの利用３　」

東京情報クリエイター

ベクトルの利用３

◆カプセルと円の衝突判定

カプセルとは以下のような形状として、円との衝突判定を考えてみます。
この判定はすでに求めた円と線分の衝突判定から簡単に求めることができます。

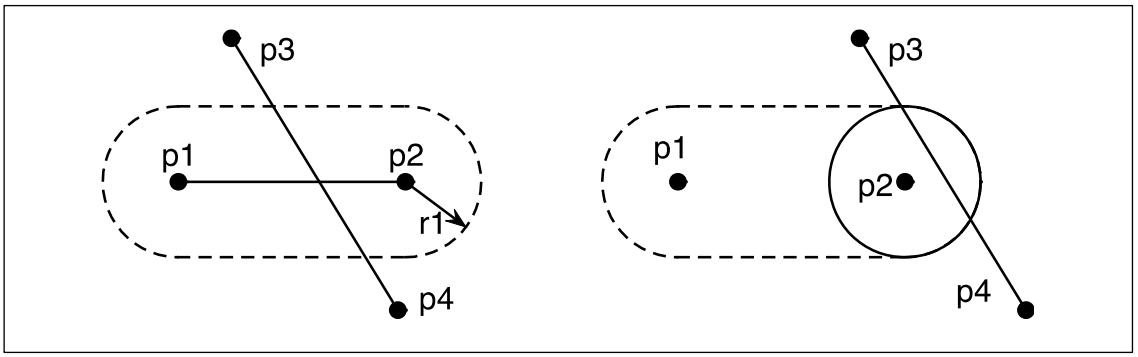


つまり、それぞれの半径の輪を円の半径として、線分と円の衝突判定を行えばよいですね。

→　Circle_Segment(　　)

◆カプセルと線分の衝突判定

今度はカプセルと線分との衝突判定を考えてみます。これは、線分の交差および円と線分の衝突を使用すれば簡単に求めることができます。



線分と線分　LineIntersection(　　)

円と線分　　Circle_Segment(　　)

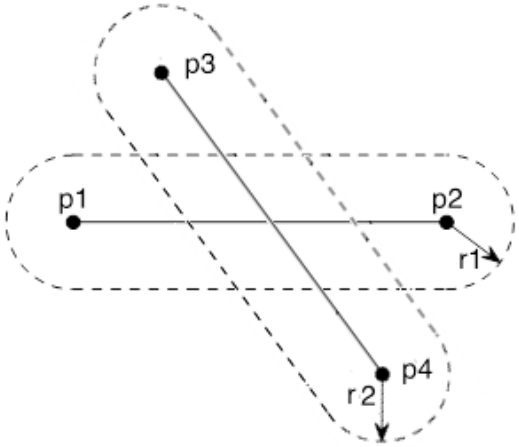
　　Circle_Segment(　　)

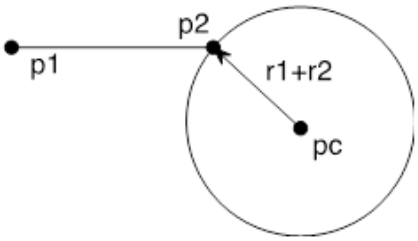
「　ベクトルの利用３　」

東京情報クリエイター

◆カプセルとカプセルの衝突判定

カプセルとカプセルの衝突を考えましょう。





この判定は、カプセルと円・カプセルと線分と同じように、線分の交差および円と線分の衝突を使用すれば簡単に求めることができますね。

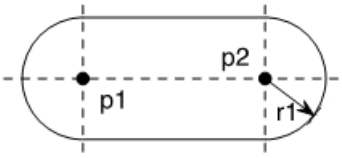
線分と線分	LineIntersection(	)
円と線分	Circle_Segment(	)
	Circle_Segment(	)
	Circle_Segment(	)
	Circle_Segment(	)

◆演習問題1

上記を利用し、Collision2D.cs に[Capsule_Circle]メソッドと[Capsule_Segment]メソッド[Capsule_Capsule]メソッドを作成しなさい。戻り値はbool とする。

プロジェクト名[Math03Ex06] [Math03Ex07] [Math03Ex08]をテンプレートで新規に作成する。
描画が完成したら、アニメーションもつけてみましょう。

カプセルの描画




```
r1=20;
SetThick(r1 * 2);
Line(p1, p2);
```

「　ベクトルの利用３　」

東京情報クリエイター

[Collision2D.cs 追加]

```
    // <summary>
    // カプセルと円の衝突判定
    // </summary>
    // <param name="p1">カプセル中心線端点１</param>
    // <param name="p2">カプセル中心線端点１</param>
    // <param name="r1">カプセル半径</param>
    // <param name="pc">円の中心</param>
    // <param name="r2">円の半径</param>
    // <returns></returns>
    public static bool Capsule_Circle
        (Vector2 p1, Vector2 p2, float r1, Vector2 pc, float r2)
    {
        float radius = r1 + r2;
        if (Circle_Segment(ref pc, radius, p1, p2))  return true;
        return false;
    }

    // <summary>
    // カプセルと線分の衝突判定
    // </summary>
    // <param name="p1">カプセル中心線端点１</param>
    // <param name="p2">カプセル中心線端点１</param>
    // <param name="r">カプセル半径</param>
    // <param name="p3">線分端点１</param>
    // <param name="p4">線分端点２</param>
    // <returns></returns>
    public static bool Capsule_Segment (Vector2 p1,Vector2 p2,float r,Vector2 p3,Vector2 p4)
    {
        bool result1 = LineIntersection(p1, p2, p3, p4);
        bool result2 = Circle_Segment(ref p1, r, p3, p4);
        bool result3 = Circle_Segment(ref p2, r, p3, p4);

        return result1 || result2 || result3;
    }

    // <summary>
    // カプセルとカプセル
    // </summary>
    // <param name="p1">カプセルａ中心線端点１</param>
    // <param name="p2">カプセルａ中心線端点２</param>
    // <param name="r1">カプセルａ半径</param>
    // <param name="p3">カプセルｂ中心線端点１</param>
    // <param name="p4">カプセルｂ中心線端点２</param>
    // <param name="r2">カプセルａ半径</param>
    // <returns></returns>
    public static bool Capsule_Capsule
        (Vector2 p1,Vector2 p2,float r1,Vector2 p3,Vector2 p4,float r2)
    {
        float radius = r1 + r2;

        //中心線同士が交差してるか？
        if (LineIntersection(p1, p2, p3, p4)) return true;

        //カプセルの端点（ｒの中心点）が線分に交差しているか？
        if (Circle_Segment(p1, radius, p3, p4)) return true;
        if (Circle_Segment(p2, radius, p3, p4)) return true;
        if (Circle_Segment(p3, radius, p1, p2)) return true;
        if (Circle_Segment(p4, radius, p1, p2)) return true;

        return false;
    }
```

「　ベクトルの利用３　」

東京情報クリエイター

[Math03Ex06　MyDrawClass.cs] カプセルと円の衝突判定　描画のみ

```
namespace Math03Ex06
{
    public class MyDrawClass:Draws,IDraws
    {
        InputState input;
        Vector2 p0, p1, p2, p3;    //★修正
        float r1, r2;    //★追加

        public MyDrawClass()
        { //☆修正なし }

        public void InputData()
        {
            Init(); //属性の初期化
            Clear();//描画領域のクリア（座標軸のみ描画されます）

            List<PointF> p = new List<PointF>();
            p = input.GetPoint(2, "円の中心と円周上の点の２点を入力");  //★修正
            if (p.Count == 0) { Clear(); return; }//入力キャンセル処理

            p0 = new Vector2(p[0].X, p[0].Y);  //円の中心

            //★以降を追加
            p1 = new Vector2(p[1].X, p[1].Y);  //円周上の点
            r1 = (p1 - p0).Length();  //円の半径
            Circle(p0, r1);
            Render(); //円の描画；

            p = input.GetPoint(2, "カプセル円弧中心の２点を入力");
            if (p.Count == 0) { Clear(); return; }//入力キャンセル処理
            p2 = new Vector2(p[0].X, p[0].Y);   //カプセル円弧の中心１
            p3 = new Vector2(p[1].X, p[1].Y);   //カプセル円弧の中心２
            r2 = 20;  //カプセル円の半径

            //交差判定
            if (Collision2D.Capsule_Circle(p2, p3, r2, p0, r1))
            {
                Text(new Vector2(-200, 100), "交差しました。");
                SetColor(Color.Red);
            }
            else
            {
                Text(new Vector2(-200, 100), "交差してません。");
                SetColor(Color.RoyalBlue);
            }

            //カプセルの描画
            SetThick(r2 * 2);
            Line(p2, p3);
            Render();
        }

        public void Update() { }
        public void Draw() { }
    }
}
```

「　ベクトルの利用３　」

東京情報クリエイター

[Math03Ex06　MyDrawClass.cs] カプセルと円の衝突判定 [MyDrawClass.cs]に追加(アニメーション)

```
//★★フィールドに追加
Vector2 mov0, mov2, mov3;   //速度
bool IntersectionFlag; //交差フラグ

//★★コンストラクタに追加　(初速度は任意)
IntersectionFlag = false;
mov0 = new Vector2(1, 2);    //p0
mov2 = new Vector2(-1, -1);  //p2
mov3 = new Vector2(1, -1);   //p3

public void InputData()
{   //修正なし　}

//★★座標点の移動処理を追加
private void AreaMove(ref Vector2 position, ref Vector2 mov,float radius)
{
    if ((position.X - radius < -Width / 2) || (Width / 2 < position.X + radius)) mov.X *= -1;
    if ((position.Y - radius < -Height / 2) || (Height / 2 < position.Y + radius)) mov.Y *= -1;

    position += mov;
}

public void Update()  //★★修正
{
    AreaMove(ref p0, ref mov0, r1);    //p0 の移動範囲
    AreaMove(ref p2, ref mov2, r2);    //p2 の移動範囲
    AreaMove(ref p3, ref mov3, r2);    //p3 の移動範囲

    //交差判定
    IntersectionFlag = Collision2D.Capsule_Circle(p2, p3, r2, p0, r1);
}

public void Draw()  //★★修正
{
    //交差判定
    if (IntersectionFlag)
    {
        Text(new Vector2(-200, 100), "交差しました。");
        SetColor(Color.Red); //色を変えると以降その色のまま
    }
    else
    {
        Text(new Vector2(-200, 100), "交差してません。");
        SetColor(Color.RoyalBlue);
    }

    SetThick(1);
    Circle(p0, r1);

    SetThick(r2 * 2);
    Line(p2, p3);
}
```

「　ベクトルの利用３　」

東京情報クリエイター

[Math03Ex07　MyDrawClass.cs] カプセルと線分　描画のみ

```
//カプセルと線分の衝突判定
namespace Math03Ex07
{
    public class MyDrawClass:Draws,IDraws
    {
        InputState input;
        Vector2 p0, p1, p2, p3;  //★修正
        float r;            //★追加

        public MyDrawClass()
        {  //☆変更なし  }

        public void InputData()
        {
            Init(); //属性の初期化
            Clear();//描画領域のクリア（座標軸のみ描画されます）

            List<PointF> p = new List<PointF>();
            p = input.GetPoint(2, "線分の端点を入力");   //★修正
            if (p.Count == 0) { Clear(); return; }//入力キャンセル処理

            p0 = new Vector2(p[0].X, p[0].Y);  //線分の端点１

            //★以降追加
            p1 = new Vector2(p[1].X, p[1].Y);  //線分の端点２
            Line(p0, p1);
            Render(); //線分の描画；

            p = input.GetPoint(2, "カプセル円弧中心の２点を入力");
            if (p.Count == 0) { Clear(); return; }//入力キャンセル処理
            p2 = new Vector2(p[0].X, p[0].Y);   //カプセル円の中心１
            p3 = new Vector2(p[1].X, p[1].Y);   //カプセル円の中心２
            r = 20;  //カプセル円の半径

            //交差判定
            if (Collision2D.Capsule_Segment(p2, p3, r, p0, p1))
            {
                Text(new Vector2(-200, 100), "交差しました。");
                SetColor(Color.Red);
            }
            else
            {
                Text(new Vector2(-200, 100), "交差してません。");
                SetColor(Color.RoyalBlue);
            }

            SetThick(r * 2);
            Line(p2, p3);
            Render();
        }

        public void Update() { }
        public void Draw() { }
    }
}
```

「　ベクトルの利用３　」

東京情報クリエイター

[Math03Ex07　MyDrawClass.cs] カプセルと線分　[MyDrawClass.cs]に追加（アニメーション）

```
//★★フィールドに追加
Vector2 mov0, mov1, mov2, mov3;
bool IntersectionFlag;

//★★コンストラクタに追加
IntersectionFlag = false;
mov0 = new Vector2(1, 2);    //p0
mov1 = new Vector2(1, 1);    //p1
mov2 = new Vector2(-1, -1);  //p2
mov3 = new Vector2(1, -1);   //p3

public void InputData()
{   //修正なし   }

//★★座標点の移動処理を追加
private void AreaMove(ref Vector2 position, ref Vector2 mov, float radius)
{
    if ((position.X - radius < -Width / 2) || (Width / 2 < position.X + radius)) mov.X *= -1;
    if ((position.Y - radius < -Height / 2) || (Height / 2 < position.Y + radius)) mov.Y *= -1;

    position += mov;
}

public void Update()
{
    AreaMove(ref p0, ref mov0 , 0); //p0の移動範囲
    AreaMove(ref p1, ref mov1 , 0); // p1の移動範囲
    AreaMove(ref p2, ref mov2,r);  //p2の移動範囲
    AreaMove(ref p3, ref mov3,r);  //p3の移動範囲

    //交差判定
    IntersectionFlag = Collision2D.Capsule_Segment(p2, p3, r, p0, p1);
}

public void Draw()
{
    //交差判定
    if (IntersectionFlag)
    {
        Text(new Vector2(-200, 100), "交差しました。");
        SetColor(Color.Red); //色を変えると以降その色のまま
    }
    else
    {
        Text(new Vector2(-200, 100), "交差してません。");
        SetColor(Color.RoyalBlue);
    }

    //カプセル
    SetThick(r * 2);
    Line(p2, p3);

    //線分
    SetThick(1);
    Line(p0, p1);
}
```

「　ベクトルの利用３　」

東京情報クリエイター

[Math03Ex08　MyDrawClass.cs] カプセルとカプセル

```
//カプセルとカプセル
namespace Math03Ex08
{
    public class MyDrawClass:Draws,IDraws
    {
        InputState input;
        Vector2 p0, p1, p2, p3;  //★修正
        float r1, r2;  //★追加

        public MyDrawClass()
        {  ☆変更なし  }

        public void InputData()
        {
            Init(); //属性の初期化
            Clear();//描画領域のクリア（座標軸のみ描画されます）

            List<PointF> p = new List<PointF>();
            p = input.GetPoint(2, "カプセル１円弧中心の２点を入力");  //★修正
            if (p.Count == 0) { Clear(); return; }//入力キャンセル処理

            p0 = new Vector2(p[0].X, p[0].Y);  //カプセル１円弧中心１

            //★以降追加
            p1 = new Vector2(p[1].X, p[1].Y);  //カプセル１円弧中心２
            r1 = 20;  //カプセル１円の半径
            SetThick(r1 * 2);
            Line(p0, p1);
            Render();   //カプセル１の描画

            p = input.GetPoint(2, "カプセル２円弧中心の２点を入力");
            if (p.Count == 0) { Clear(); return; }//入力キャンセル処理
            p2 = new Vector2(p[0].X, p[0].Y);  //カプセル２円弧中心１
            p3 = new Vector2(p[1].X, p[1].Y);  //カプセル２円弧中心２
            r2 = 30;  //カプセル２円の半径

            //交差判定
            if (Collision2D.Capsule_Capsule(p0, p1, r1, p2, p3,r2))
            {
                Text(new Vector2(-200, 100), "交差しました。");
                SetColor(Color.Red);
            }
            else
            {
                Text(new Vector2(-200, 100), "交差してません。");
                SetColor(Color.RoyalBlue);
            }

            SetThick(r2 * 2);
            Line(p2, p3);
            Render();  //カプセル２の描画
        }
        public void Update() { }
        public void Draw() { }
    }
}
```

「　ベクトルの利用３　」

東京情報クリエイター

[Math03Ex08 MyDrawClass.cs] カプセルとカプセル　[MyDrawClass.cs]に追加（アニメーション）

```
//★★フィールドに追加
Vector2 mov0, mov1, mov2, mov3;
bool IntersectionFlag;

//★★コンストラクタに追加
IntersectionFlag = false;
mov0 = new Vector2(1, 2);    //p0
mov1 = new Vector2(1, 1);    //p1
mov2 = new Vector2(-1, -1);  //p2
mov3 = new Vector2(1, -1);   //p3

public void InputData()
{   //修正なし   }

//★★座標点の移動処理を追加
private void AreaMove(ref Vector2 position, ref Vector2 mov, float radius)
{
    if ((position.X - radius < -Width / 2) || (Width / 2 < position.X + radius)) mov.X *= -1;
    if ((position.Y - radius < -Height / 2) || (Height / 2 < position.Y + radius)) mov.Y *= -1;

    position += mov;
}

public void Update()
{
    AreaMove(ref p0, ref mov0, r1);  //p0の移動範囲
    AreaMove(ref p1, ref mov1, r1);  // p1の移動範囲
    AreaMove(ref p2, ref mov2, r2);  //p2の移動範囲
    AreaMove(ref p3, ref mov3, r2);  //p3の移動範囲

    //交差判定
    IntersectionFlag = Collision2D.Capsule_Capsule( p0, p1, r1, p2, p3, r2);
}

public void Draw()
{
    //交差判定
    if (IntersectionFlag)
    {
        Text(new Vector2(-200, 100), "交差しました。");
        SetColor(Color.Red); //色を変えると以降その色のまま
    }
    else
    {
        Text(new Vector2(-200, 100), "交差してません。");
        SetColor(Color.RoyalBlue);
    }

    SetThick(r1 * 2);
    Line(p0, p1);

    SetThick(r2 * 2);
    Line(p2, p3);
}
```

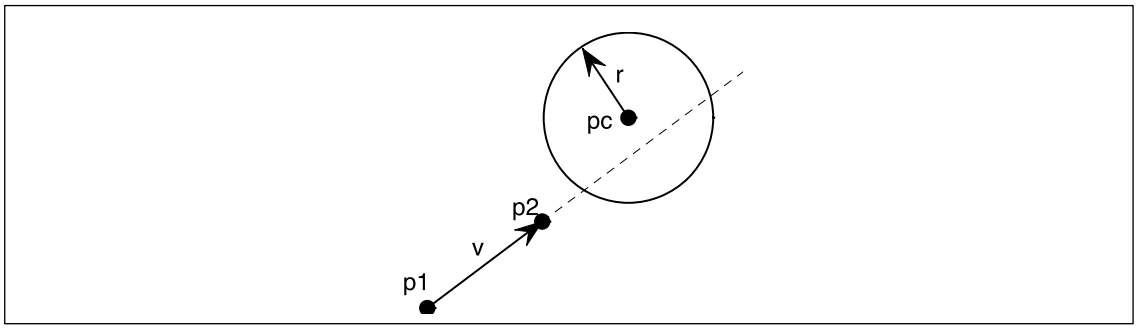
「　ベクトルの利用３　」

東京情報クリエイター

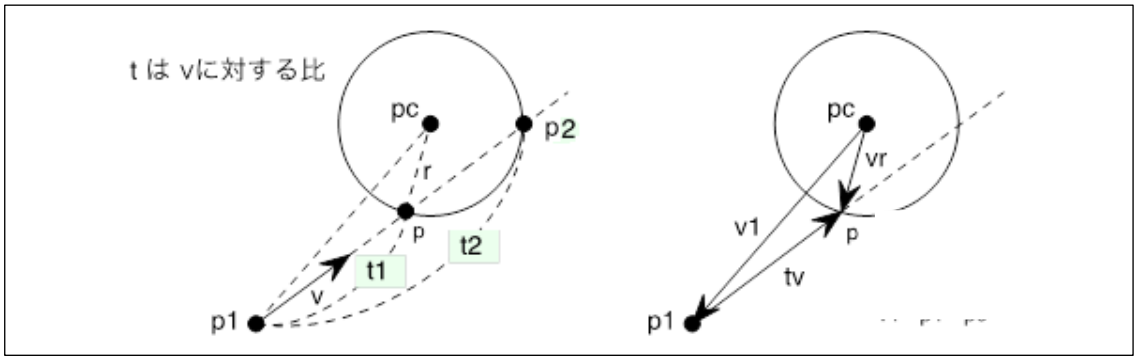
◆円とレイの衝突判定

円とレイの衝突は２Ｄゲームにおいてキャラクターに対してビームを撃った時にキャラクターに当たったかとか、視線方向に対してキャラクターがいるかどうか判定する時などに利用することができます。

最初に、円とレイを次のように定義します。



ここで、レイと円が交差する点は２点存在します。その**交点までの比を** t として考え、それぞれ**手前の交点を** t_1**、奥の交点を** t_2 とします。
次に下図右のようにベクトルに置き、t を求めることを考えます。



ベクトルの和を利用し、$\overrightarrow{v_r}$ についての式を立てます。

そして、$\overrightarrow{v_r}$ の長さ $|\overrightarrow{v_r}|$ は半径 r と同じです。あるベクトルの長さの２乗はそのベクトル同士の内積に等しいということを利用し、方程式を立ててみます。

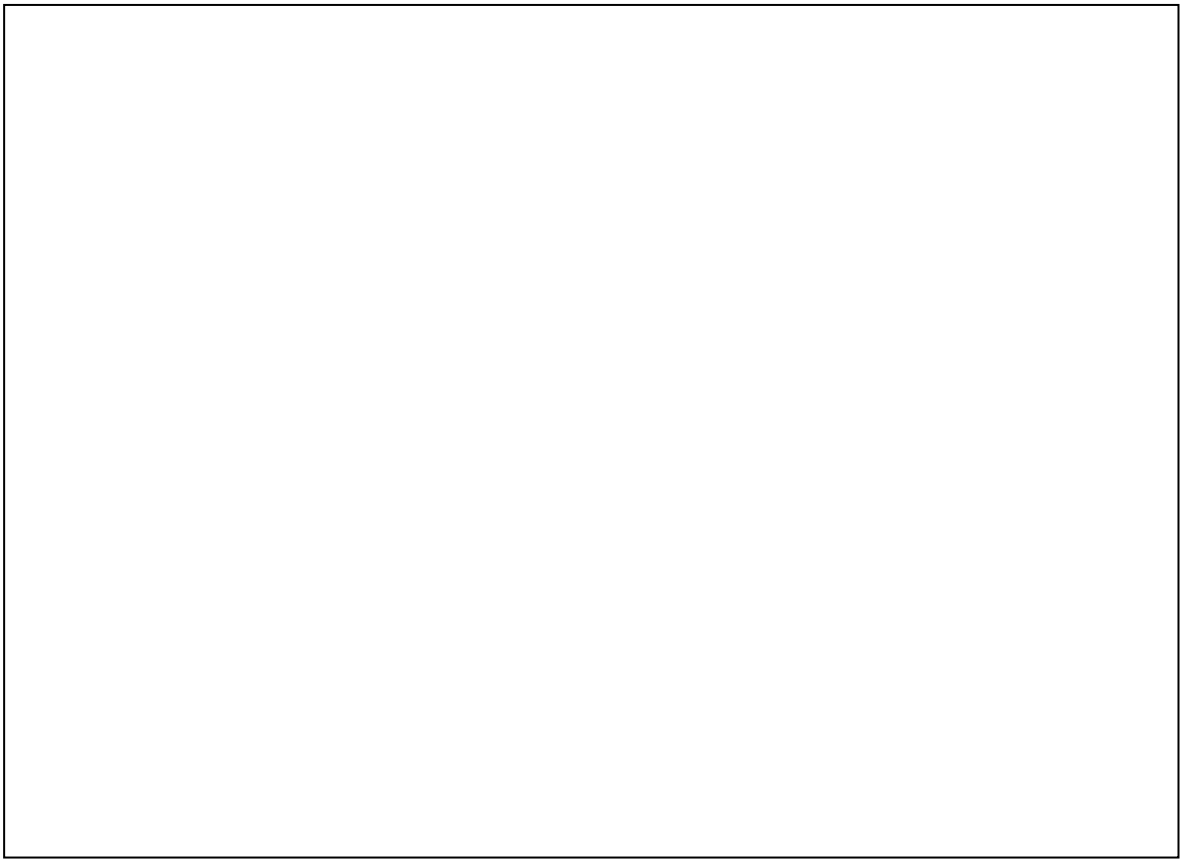
$|\overrightarrow{v_r}| =$　　$\overrightarrow{v_1} =$　　$\vec{v} =$

$\overrightarrow{v_r} =$　　$|\overrightarrow{v_r}|^2 =$

この式を展開し、t について整理すると、t についての 2 次方程式になります。

「　ベクトルの利用３　」

東京情報クリエイター



$\vec{v}$ を正規化しておけば、$\vec{v}\cdot\vec{v}=$

$t=$

①の判別式　$D'=(\overrightarrow{v_1}\cdot\vec{v})^2-(\overrightarrow{v_1}\cdot\overrightarrow{v_1}-r^2)\geqq 0$ のとき、実数解を持ちます。

つまり、レイ（直線）と円は交差することになります。よって、条件判定式は、以下のようになります。
また、交点は、t がわかれば、p_1 と $t\vec{v}$ より、次のように求めることができます。

【 条件判定式 】

のとき、衝突している

【 交点 p 】

$p=$

ただし、

$t>0$ のとき、

$t<0$ かつ $|\overrightarrow{v_1}|<r$ のとき、

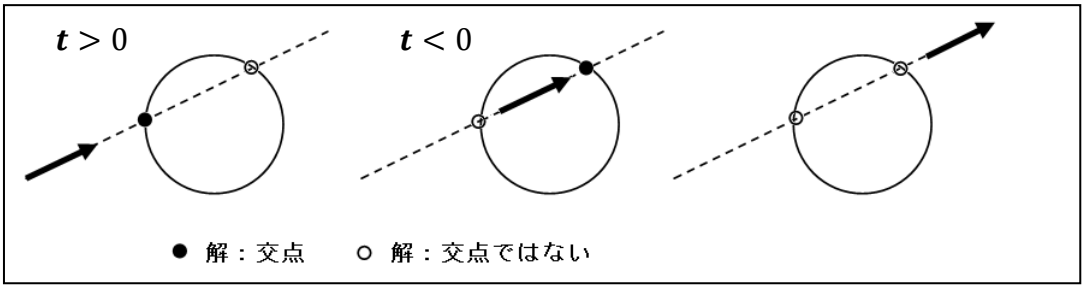
「　ベクトルの利用３　」

東京情報クリエイター

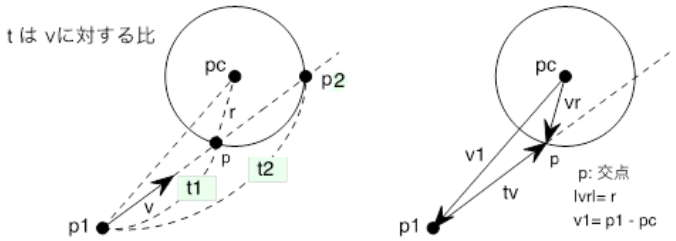
◆演習問題2

上記のことを利用して Collision2D クラスに Circle_Ray メソッドを作成しなさい。戻り値は bool とする。また、交点も得られるように考えなさい。
プロジェクト名[Math0Ex09]をテンプレートを使用して作成する。
描画が完成したら、アニメーションもつけてみましょう。

注意：レイのベクトルが境界円に向いていない時の条件も考える必要があります。
場合によっては、レイの始点が境界円に入っているようなこともあるので、そのような時はどう処理するかも考える必要もあります。



【ヒント！】



・使用する変数の定義

$$\vec{v} = \frac{p2-p1}{|p2-p1|} \qquad \overrightarrow{v_1} = p1 - pc$$

・条件判定式で使う変数の定義

$$b = \vec{v} \cdot \overrightarrow{v_1} \qquad L2 = |\overrightarrow{v_1}|^2 \quad r2 = r^2$$

・条件判定式　$D = b * b - v12 + r2$

・レイの始点位置と交点の判定（上図参照）

①$(\overrightarrow{v_1} \cdot \vec{v})^2 - (|\overrightarrow{v_1}|^2 - r^2) \geqq 0$ のとき、円とレイ（直線）は衝突している

②$t > 0$　$t = -(\overrightarrow{v_1} \cdot \vec{v}) - \sqrt{(\overrightarrow{v_1} \cdot \vec{v})^2 - (\overrightarrow{v_1} \cdot \overrightarrow{v_1} - r^2)}$

③$t < 0$　$|\overrightarrow{v_1}| < r$

$$t = -(\overrightarrow{v_1} \cdot \vec{v}) + \sqrt{(\overrightarrow{v_1} \cdot \vec{v})^2 - (\overrightarrow{v_1} \cdot \overrightarrow{v_1} - r^2)}$$

・交点　$p = p_1 + t\vec{v}$

[Collision2D.cs 追加]

```
// <summary>
// 円とレイの衝突
// </summary>
// <param name="pc">円の中心</param>
// <param name="radius">円の半径</param>
// <param name="p1">レイの始点</param>
// <param name="p2">レイの終点</param>
// <param name="Intersection"></param>
// <returns></returns>
public static bool Circle_Ray
    (Vector2 p0, float radius, Vector2 p1, Vector2 p2, ref Vector2 Intersection)
{
    Vector2 v = Vector2.Normalize(p2 - p1);
    Vector2 v1 = p1 - p0;  //レイの始点から境界円の中心までのベクトル
    float b = Vector2.Dot(v, v1); //レイの方向ベクトルと内積をとる
    float r2 = radius * radius;  //半径の２乗
    float L2 = v1.LengthSquared(); //p1 から中心までの距離の２乗
    float D = b * b - L2 + r2; //√の中が衝突判定条件となる:判別式
    float DS = (float)Math.Sqrt(D); //解の公式の√の部分
    if (D >= 0)
    {
        float t = -b - DS;
        if (t < 0)   //レイの始点位置の判定
        {
            if (L2 < r2)  //中心までの距離の２乗と半径の２乗を比較
            {
                t = -b + DS; //反対側の交点にする
            }
            else
                return false;
        }
        Intersection = p1 + t * v;
        return true;
    }
    return false;
}
```

[Math03Ex09　MyDrawClass.cs] 円とレイ

```
//円とレイの衝突判定
namespace Math03Ex09
{
    public class MyDrawClass : Draws, IDraws
    {
        InputState input;
        Vector2 p0, p1, p2, p3;  //★修正
        Vector2 Intersection;    //★追加
        float r;  //★追加

        public MyDrawClass()
        {
            input = new InputState();
            input.MouseOn();
            Intersection = new Vector2();  //★追加　交点
        }

        public void InputData()
        {
            Init(); //属性の初期化
            Clear();//描画領域のクリア（座標軸のみ描画されます）

            List<PointF> p = new List<PointF>();
            p = input.GetPoint(2, "円の中心と円周上の点の２点を入力");  //★修正
            if (p.Count == 0) { Clear(); return; }//入力キャンセル処理

            p0 = new Vector2(p[0].X, p[0].Y);  //円の中心

            //★以降追加
            p1 = new Vector2(p[1].X, p[1].Y);  //円周上の点
            r = (p1 - p0).Length();  //円の半径
            Circle(p0, r);
            Render(); //円の描画；

            p = input.GetPoint(2, "レイの２点を入力");
            if (p.Count == 0) { Clear(); return; }//入力キャンセル処理
            p2 = new Vector2(p[0].X, p[0].Y);  //レイの始点
            p3 = new Vector2(p[1].X, p[1].Y);  //レイの終点
            Line(p2, p3, 15);  //レイの表示

            //交差判定
            if (Collision2D.Circle_Ray(p0, r, p2, p3, ref Intersection))
            {
                Text(new Vector2(-200, 100), "当たり");
                SetColor(Color.Red);
                Dot(Intersection);
            }
            else
            {
                Text(new Vector2(-200, 100), "はずれ");
            }
            Render();
        }

        public void Update() { }
        public void Draw() { }
    }
}
```

[Math03Ex09　MyDrawClass.cs] 円とレイ　[MyDrawClass.cs]に追加（アニメーション）

```
//★★フィールドに追加
Vector2 mov0, mov1, mov2;
bool IntersectionFlag;

//★★コンストラクタに追加
IntersectionFlag = false;
mov0 = new Vector2(1, 2);    //p0
mov1 = new Vector2(-1, -1);  //p2
mov2 = new Vector2(1, -1);   //p3

public void InputData()
{   //修正なし   }

//★★座標点の移動処理を追加
private void AreaMove(ref Vector2 position, ref Vector2 mov, float radius)
{
    if ((position.X - radius < -Width / 2) || (Width / 2 < position.X + radius)) mov.X *= -1;
    if ((position.Y - radius < -Height / 2) || (Height / 2 < position.Y + radius)) mov.Y *= -1;

    position += mov;
}

public void Update()
{
    AreaMove(ref p0, ref mov0, r);    //p0の移動範囲

    //マウス座標の取得
    float[] p = input.MousePoistion();
    p3 = new Vector2(p[0], p[1]);

    //交差判定
    IntersectionFlag = Collision2D.Circle_Ray(p0, r, p2, p3, ref Intersection);
}

public void Draw()
{
    //交差判定
    if (IntersectionFlag)
    {
        Text(new Vector2(-200, 100), "当たり");
        SetColor(Color.Red); //色を変えると以降その色のまま
        Dot(Intersection);   //交点の動画
    }

    Circle(p0, r);
    SetColor(Color.Black);
    Line(p2, p3, 15);
}
```