「　ベクトルの利用１　」

東京情報クリエイター

# ベクトルの利用１

## ◆点と直線の距離を求める。（点と直線の最短距離を求める）

いま図 1-1 のような $\boldsymbol{p_1}$ $\boldsymbol{p_2}$ を通る直線を考え、任意点 $\boldsymbol{p_0}$ から直線へ垂線を下ろした時の距離を $\boldsymbol{L}$ とします。

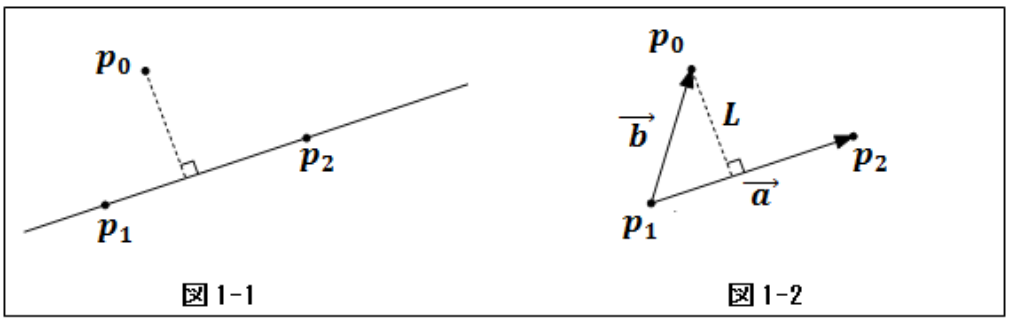


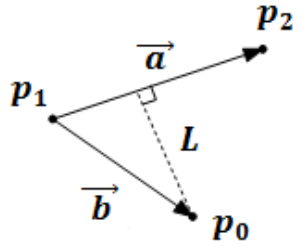

図 1-1　　図 1-2

そこで、図 1-2 のように、$\overrightarrow{p_1p_2}$ をベクトル $\vec{a}$、$\overrightarrow{p_1p_0}$ をベクトル $\vec{b}$ と考えれば、以下のように $\boldsymbol{L}$ を２つのベクトルの外積として求めることができます。

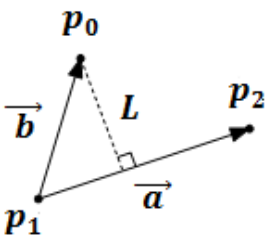
$\boldsymbol{L} =$　　$\dfrac{\vec{a}}{|\vec{a}|} = \vec{A}$　　とすると、$\boldsymbol{L} =$

すなわち、$\vec{a}$ は $\boldsymbol{p_1}$ を始点とする単位ベクトルを使用すれば、正規化をすることなく $\boldsymbol{L}$ を求めることができるのが分かります。ただし、上式は次の場合は符号が逆になるので、注意をする必要があります。また、外積をする順番が逆の場合も同様です。

次のような場合、$L < 0$ となる




$L = \dfrac{\vec{a}}{|\vec{a}|} \times \vec{b}$




$L = \vec{b} \times \dfrac{\vec{a}}{|\vec{a}|}$

**【例題１】**

2 点 $\mathbf{A(1,2)}$　$\boldsymbol{B(-2,14)}$ を通る直線と点 $\boldsymbol{C(-4,5)}$の最短距離を求めなさい。

**【解答】**

$\overrightarrow{AB} =$

$\overrightarrow{AC} =$

$|\overrightarrow{AB}| =$

$\overrightarrow{AB}$ の正規化 $(\vec{a}) = \dfrac{\overrightarrow{AB}}{|\overrightarrow{AB}|} =$

$\vec{a} \times \overrightarrow{AC} =$

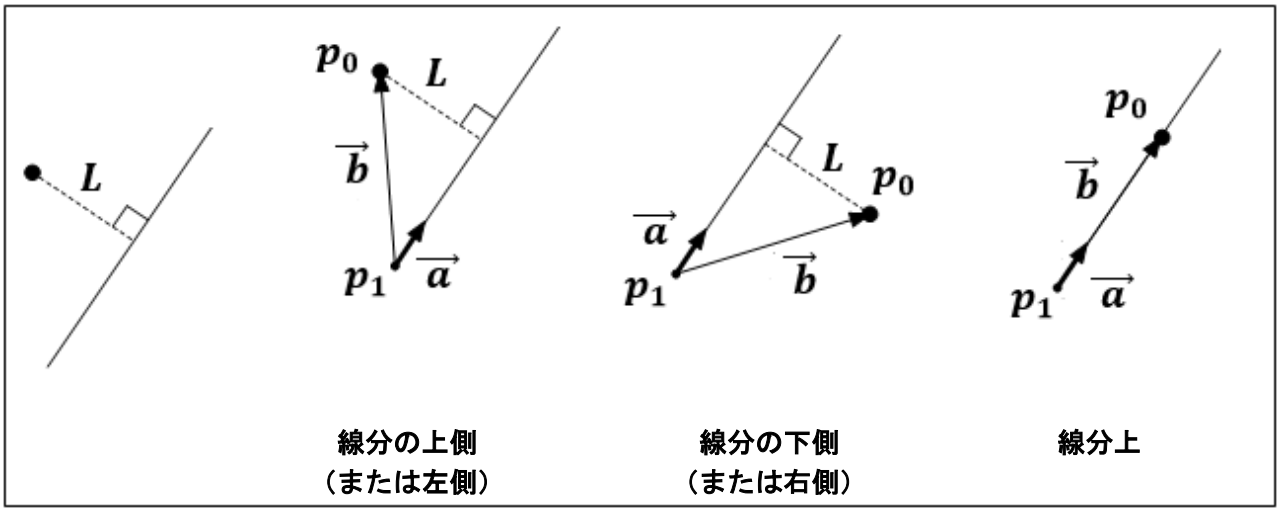
「　ベクトルの利用１　」

東京情報クリエイター

また、$L$ の符号によって、直線に対して任意点 $p_0$ がどちら側にあるかを判別することもできます。

**◆任意点が直線上のどこか調べる**

下図のように任意点が線上のどこにあるかを距離（外積）から判断することができます。



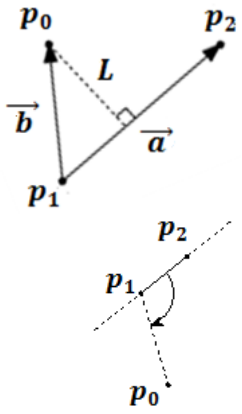

プログラミングにおいて $L=0$ を満たす状態はほとんどありません。
そこで、線分上（すなわち $L=0$ ）を見つける条件式は次のようにしなければなりません。

$$|L|<\varepsilon$$

ここで、$\varepsilon$ は計算誤差とし、適当な値（例えば $0.001$ など）を設定して誤差の範囲内になったら$0$と判断するようにします。

**【例題２】**

$p_1(6,10)$ , $p_2(11,12)$を通る直線があります。点 $p_0$ が $(10,5)$ のとき、$p_0$ は直線に対してどちら側にあるか答えなさい。




外積を使い、$L$ の値の符号を見れば良い。

$\vec{a}=p_2-p_1=$

$\vec{b}=p_0-p_1=$

$L=\vec{a}\times\vec{b}\ =$




$L>0$ のとき反時計側、$L<0$ のとき時計側

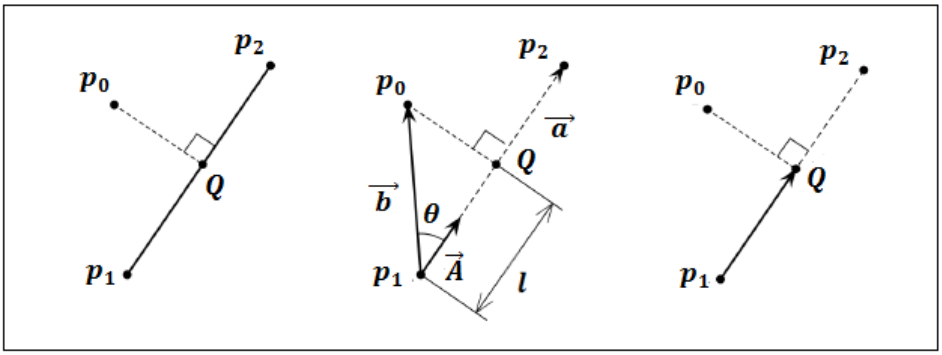
よって、点 $p_0$ は直線に対して $p_1$ を回転中心として　　。

「　ベクトルの利用１　」

東京情報クリエイター

## ◆任意点から直線に垂線を下ろした交点

図のような直線 $\boldsymbol{p_1p_2}$ に任意点 $\boldsymbol{p_0}$ から推薦を下ろした時の交点 $\boldsymbol{Q}$ を求めるために、ベクトル $\overrightarrow{\boldsymbol{p_1p_2}}$ とベクトル $\overrightarrow{\boldsymbol{p_1p_0}}$ を考え、それぞれ $\overrightarrow{\boldsymbol{a}}, \overrightarrow{\boldsymbol{b}}$ とします。




この 2 つのベクトルを利用し、ベクトル $\overrightarrow{\boldsymbol{a}}$ を正規化した $\overrightarrow{\boldsymbol{A}}$ と $\overrightarrow{\boldsymbol{b}}$ の内積を取れば、$\boldsymbol{p_1}$ から $\boldsymbol{Q}$ までの距離 $\boldsymbol{l}$ を求めることができます。

$$\boldsymbol{l} =$$

この長さを利用し、ベクトル $\overrightarrow{\boldsymbol{a}}$ を正規化した $\overrightarrow{\boldsymbol{A}}$ と $\boldsymbol{l}$ を掛ければベクトル $\overrightarrow{\boldsymbol{p_1Q}}$ を求めることができます。

$$\overrightarrow{\boldsymbol{p_1Q}} = \boldsymbol{l}\,\overrightarrow{\boldsymbol{A}} \quad =$$

よって $\boldsymbol{Q}$ 点の座標値は次のようになります。　$\boldsymbol{Q} =$

## ◆ベクトルの利用　プログラム1

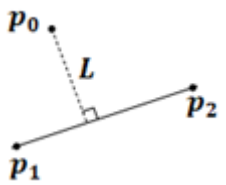
**１．点と直線の距離を求めるプログラムを作成しましょう。**

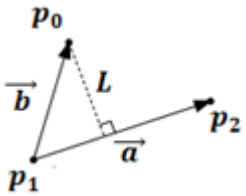
**(1)[MathCalc]ソリューション内にテンプレートを使用して新しいプロジェクトを作成する。**
**（プロジェクト名　[Math03Ex02]）**

**(2)[Math03Ex02]をスタートアップ プロジェクトに設定**

**【ヒント！】**

**①入力：3 点**　　**②ベクトル $\overrightarrow{\boldsymbol{a}}, \overrightarrow{\boldsymbol{b}}$ を求める**　　**③$\overrightarrow{a}$ を正規化する**

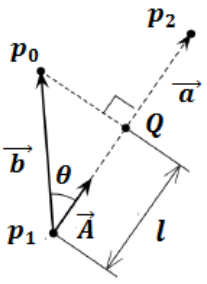



$$\vec{\boldsymbol{A}} = \frac{\vec{\boldsymbol{a}}}{|\vec{\boldsymbol{a}}|}$$

**④ベクトル $\overrightarrow{\boldsymbol{A}}, \overrightarrow{\boldsymbol{b}}$ の外積(距離$\boldsymbol{L}$)を求める**　　**⑤結果を描画する。**

$$\boldsymbol{L} = \left|\overrightarrow{\boldsymbol{A}} \times \overrightarrow{\boldsymbol{b}}\right|$$




描画するためには、$\boldsymbol{p_0}$ から $\boldsymbol{p_1p_2}$ に垂線を下ろした時の交点（$\boldsymbol{Q}$ ）を求める必要がありますね。

$$\boldsymbol{l} = \overrightarrow{\boldsymbol{A}} \cdot \overrightarrow{\boldsymbol{b}} \qquad \overrightarrow{\boldsymbol{p_1Q}} = \boldsymbol{l}\,\overrightarrow{\boldsymbol{A}}$$

$$\boldsymbol{Q} = \boldsymbol{p_1} + \overrightarrow{\boldsymbol{p_1Q}} = \boldsymbol{p_1} + \boldsymbol{l}\,\overrightarrow{\boldsymbol{A}}$$

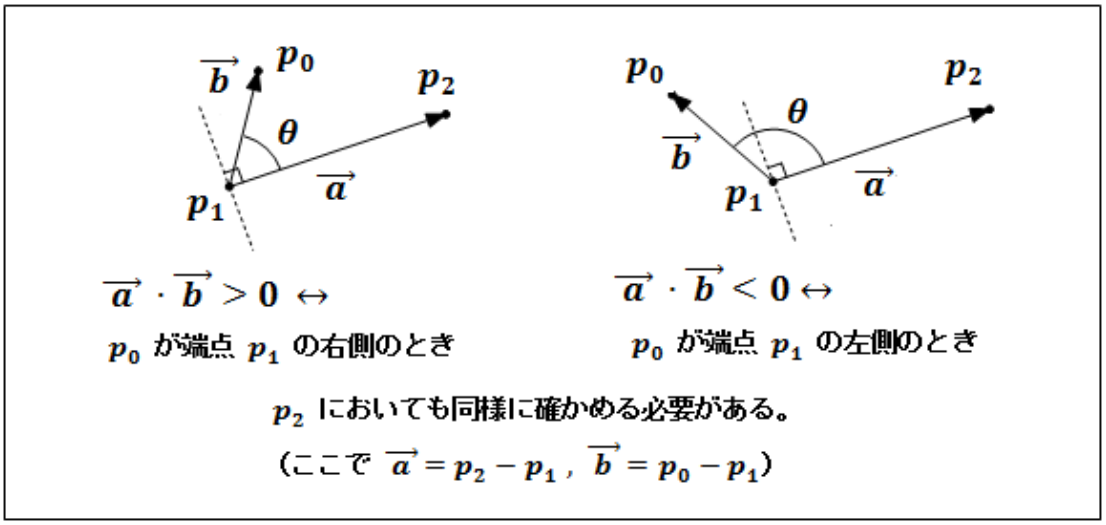
「　ベクトルの利用１　」

東京情報クリエイター

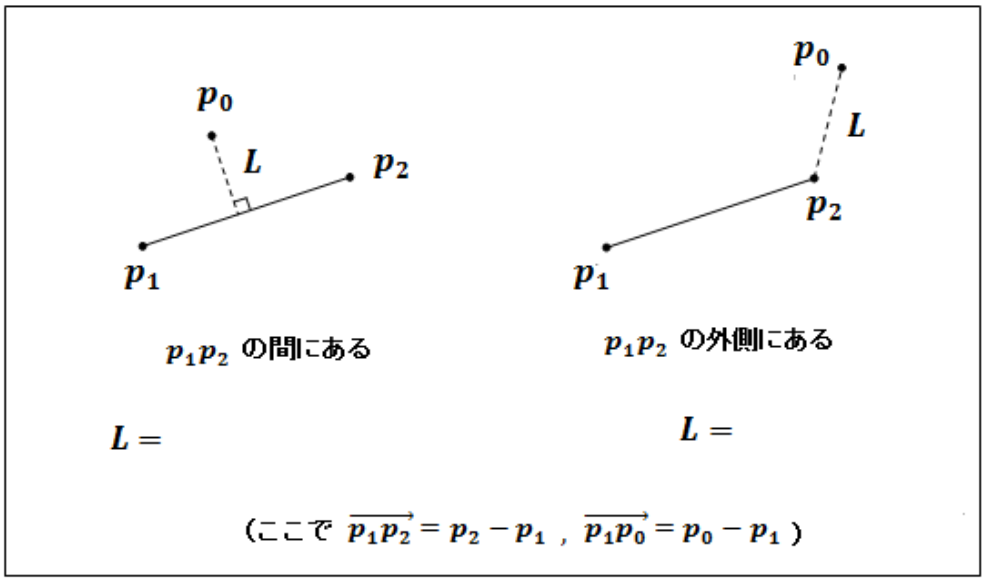
## ◆点と線分との距離を求める

前記で説明した点と直線の距離はあくまで無限の長さを持つ直線に対して垂線を下ろした時の距離です。ここでは、端点を持つ線分に対して最短距離を求めることを考えます。

任意点から下ろした垂線が線分上にない場合は、近い方の端点までの距離を最短距離にします。
そのためにまず、任意点から下ろした垂線が端点上にあるかどうかを調べます。これは、内積を使って確かめることができます。




これを両端点で確かめ端点の内側にある時だけ外積を使って距離を求め、それ以外は近い端点までの距離（２点間の距離）を求めればよいことになります。



「　ベクトルの利用１　」

東京情報クリエイター

**【点と直線の距離を求める　MyDrawClass.cs　プログラム例】**

```
namespace Math03Ex02
{
    public class MyDrawClass : Draws, IDraws
    {
        //フィールド
        InputState input;
        Vector2 p0, p1, p2;  //★修正マウス入力
        Vector2 a, b, A, Q;  //★追加
        float L, l;          //★追加

        //コンストラクタ
        public MyDrawClass()
        {
            //変更なし
        }

        public void InputData()
        {
            Init();
            Clear();
            List<PointF> p = new List<PointF>();
            p = input.GetPoint(3, "点と直線の端点を入力");  //★修正
            if (p.Count == 0) { Clear(); return; }
            p0 = new Vector2(p[0].X, p[0].Y);
            //★以下追加
            p1 = new Vector2(p[1].X, p[1].Y);
            p2 = new Vector2(p[2].X, p[2].Y);

            a = p2 - p1;
            b = p0 - p1;
            A = Vector2.Normalize(a);

            L = Math.Abs(Vector2.Cross(A, b));
            l = Vector2.Dot(A, b);
            Q = p1 + l * A;

            Draw();
            Render();
        }

        public void Update() {  }

        public void Draw()      //主に描画処理を記述
        {
            Dot(p1,5);
            Dot(p2,5);
            Line(p1, p2);
            Text(p1, "p1" + p1.ToString());
            Text(p2, "p2" + p2.ToString());

            SetColor(Color.Blue);
            Dot(p0,5);
            Text(p0, "p0" + p0.ToString());

            Dot(Q,5);
            Text(Q, "Q" + Q.ToString());
            Line(p0, Q);

            Vector2 mid = (p0 + Q) / 2;
            Text(mid, "距離：" + L);
        }
    }
}
```

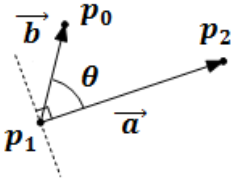
「　ベクトルの利用１　」

東京情報クリエイター

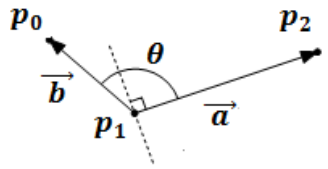
**【例題３】**

$\boldsymbol{p_1(6,10)}$，$\boldsymbol{p_2}$ $\boldsymbol{(11,12)}$ を端点とする線分があります。点 $\boldsymbol{p_0}$ が $\boldsymbol{(2,3)}$ の時、点と線分の最短距離を求めなさい。

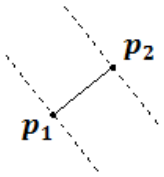
点 $\boldsymbol{p_0}$ が線分の端点を境（破線）にどちら側にあるかを調べには、内積を用いる。




$\vec{a} \cdot \vec{b} > 0 \leftrightarrow \theta < 90°$




$\vec{a} \cdot \vec{b} < 0 \leftrightarrow \theta > 90°$

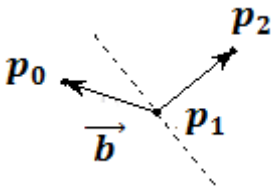



まず、$\boldsymbol{p_1}$ から調べる。

$\vec{a} = p_2 - p_1 =$

$\vec{b} = p_0 - p_1 =$

$\vec{a} \cdot \vec{b} =$

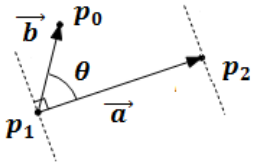


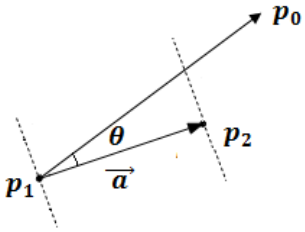

$\boldsymbol{p_0}$ は $\boldsymbol{p_1}$ の外側（ $\boldsymbol{p_1}$ から見て反時計回り側）にあるので、端点 $\boldsymbol{p_2}$ は調べる必要がない。

次に外側にあるので、最短距離は、$\boldsymbol{p_1p_0}$ の２点間距離、すなわちベクトル $\vec{b}$ の大きさ（長さ）である。

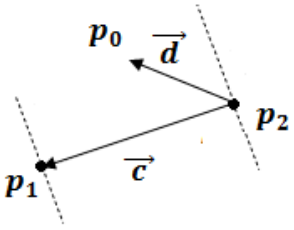
$L = |\vec{b}| =$

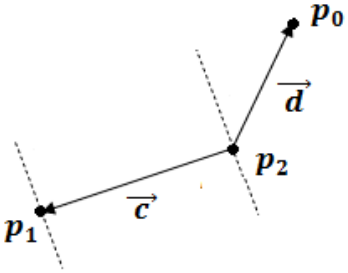
★この問題では、$\boldsymbol{p_0}$ は $\boldsymbol{p_1}$ の外側にあったので、端点 $\boldsymbol{p_2}$ は調べる必要がなかったが、$\boldsymbol{p_0}$ が $\boldsymbol{p_1}$ の内側にあった場合、以下の場合が考えられます。




このため、次に　$\vec{c} = p_1 - p_2 (= -\vec{a})$ と $\vec{d} = p_0 - p_2$ の内積を調べ必要があります。




$\vec{c} \cdot \vec{d} > 0 \left(-\vec{a} \cdot \vec{d} > 0\right) \leftrightarrow \theta < 90°$　　$\vec{c} \cdot \vec{d} < 0 \left(-\vec{a} \cdot \vec{d} > 0\right) \leftrightarrow \theta > 90°$

「　ベクトルの利用１　」

東京情報クリエイター

## ◆ベクトルのプログラム2

点と線分の距離を求めるプログラムを作成しましょう。

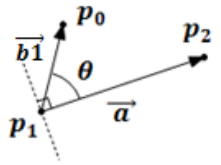
(1)[MathCalc]ソリューション内にテンプレートを使用して新しいプロジェクトを作成する。
（プロジェクト名　[Math03Ex03]）

(2)[Math03Ex03]をスタートアップ プロジェクトに設定

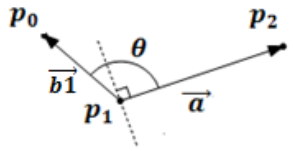
【ヒント！】

①入力：3 点

②内積を求め、$p_1$ との位置関係を調べる



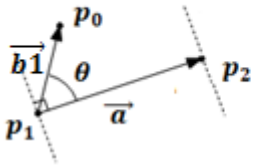

$\vec{a}\cdot\overrightarrow{b1}>0 \leftrightarrow \theta<90°$

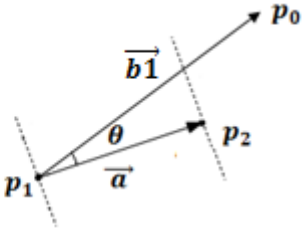


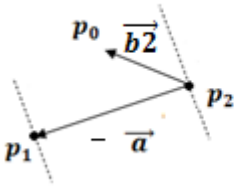

$\vec{a}\cdot\overrightarrow{b1}<0 \leftrightarrow \theta>90°$

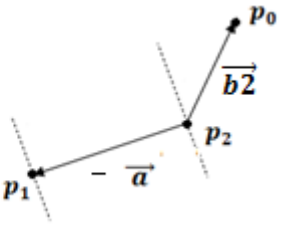
③$p_1$ の外側の場合、$p_1p_0$ の距離を計算する。

④$p_0$ が内側の場合、$p_2$ との位置関係を調べる。



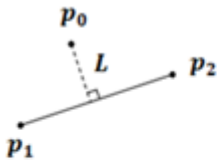



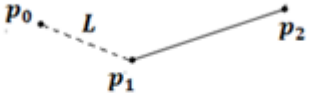

$\overrightarrow{p2p1}\cdot\overrightarrow{p0p2}=-\vec{a}\cdot\overrightarrow{b2}<0 \leftrightarrow \theta>90°$

⑤$p_2$ の外側の場合、$p_2p_0$ の距離を計算する。

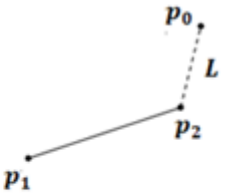
⑥ともに内側の場合、外積を求め直線との距離を求める。




$p_1p_2$ の間にある




$p_1p_2$ の外側にある




$p_1p_2$ の外側にある

$$L=\left|\frac{\overrightarrow{p_1p_2}}{|\overrightarrow{p_1p_2}|}\times\overrightarrow{p_1p_0}\right|=\left|\vec{A}\times\overrightarrow{b1}\right| \qquad L=|\overrightarrow{p_1p_0}|=|\overrightarrow{b1}| \qquad L=|\overrightarrow{p_2p_0}|=|\overrightarrow{b2}|$$

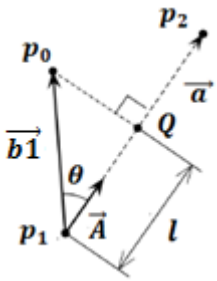
（ここで $\overrightarrow{p_1p_2}=\vec{a}$，$\overrightarrow{p_1p_0}=\overrightarrow{b1}$，$\vec{A}=\frac{\vec{a}}{|\vec{a}|}$ ）

⑦結果を描画する

描画するためには、$p_0$ から $p_1p_2$ に垂線を下ろした時の交点（$Q$ ）を求める必要がありますね。

$$l=\overrightarrow{A}\cdot\overrightarrow{b1} \qquad \overrightarrow{p_1Q}=l\overrightarrow{A}$$

$$Q=p_1+\overrightarrow{p_1Q}=p_1+l\overrightarrow{A}$$



【　メ　　モ　】

「　ベクトルの利用１　」

東京情報クリエイター

**【点と線分の距離 MyDrawClass.cs プログラム例】**

```
namespace Math03Ex03
{
    public class MyDrawClass : Draws, IDraws
    {
        InputState input;
        Vector2 p0, p1, p2;      //★修正
        Vector2 a, b1, b2, A;    //★追加
        float l, L;              //★追加
        Vector2 m, mid;          //★追加　描画で使用

        public MyDrawClass()
        {
                //変更なし
        }

        public void InputData()
        {
            Init();
            Clear();
            List<PointF> p = new List<PointF>();
            p = input.GetPoint(3, "点と線分の端点を入力");  //★修正
            if (p.Count == 0) { Clear(); return; }
            p0 = new Vector2(p[0].X, p[0].Y);
            //★以下追加
            p1 = new Vector2(p[1].X, p[1].Y);
            p2 = new Vector2(p[2].X, p[2].Y);

            a = p2 - p1;
            b1 = p0 - p1;
            b2 = p0 - p2;
            A = Vector2.Normalize(a);

            if(Vector2.Dot(a,b1)<0)  //p1 の外側
            {
                L = Vector2.Length(b1);
                m = p1;
            }
            else if(Vector2.Dot(-a,b2)<0)   //p2 の外側
            {
                L = Vector2.Length(b2);
                m = p2;
            }
            else  //p1p2 の内側
            {
                L = Math.Abs(Vector2.Cross(A, b1));
                l = Vector2.Dot(A, b1);
                m = p1 + l * A;
            }

            Draw();
            Render();
        }

        public void Update() {  }
```

「　ベクトルの利用１　」

東京情報クリエイター

```
        public void Draw()      //主に描画処理を記述
        {
            Dot(p0);
            Text(p0, "p0" + p0.ToString());
            Dot(p1);
            Text(p1, "p1" + p1.ToString());
            Dot(p2);
            Text(p2, "p2" + p2.ToString());

            Line(p1, p2);
            Line(p0, m);
            mid = (p0 + m) / 2;
            Text(mid, "距離：" + L);
        }
    }
}
```