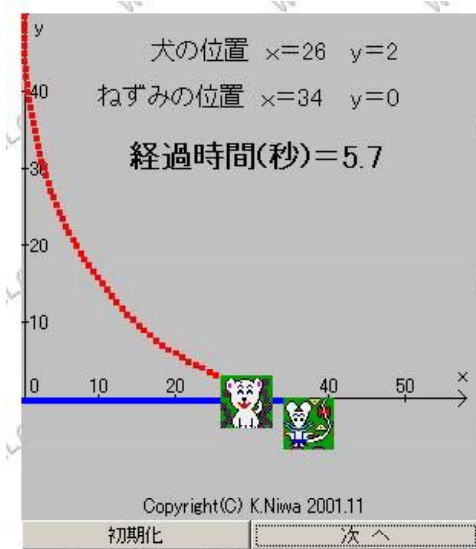
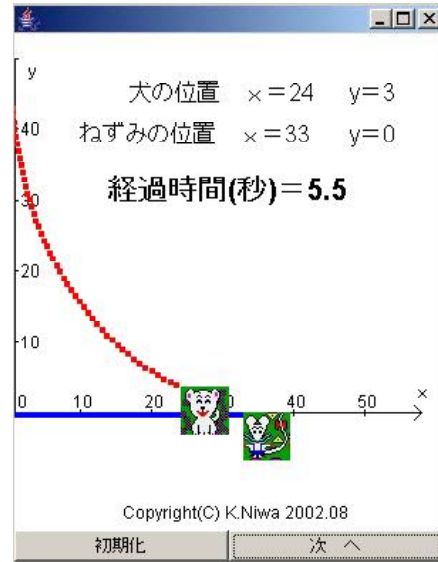


## 【ねずみを追う犬】

## 1. はじめに



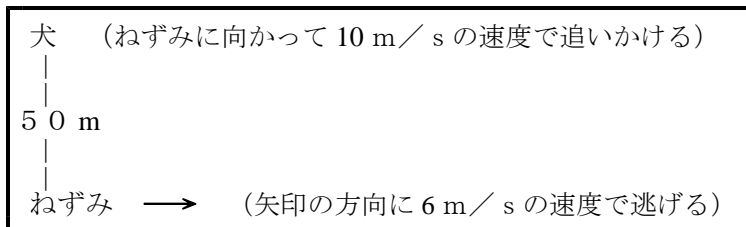
[ J a v a ア プ レ ッ ト ]



[ J a v a アプリケーション ]

【問題】

下図のように、ねずみと犬は最初  $50\text{ m}$  離れた位置にいます。  
ねずみは、まっすぐに矢印の方向に  $6\text{ m/s}$  の速度で逃げます。  
犬は、ねずみに向かってまっすぐに  $10\text{ m/s}$  の速度で追いかけます。  
ねずみを追いかける犬はどんな道筋をたどるのでしょうか。  
また、犬はねずみに何秒後に追いつくのでしょうか。



シミュレーションソフト「ねずみを追う犬」を使って、ねずみを追う犬の様子を観察してみてください。

## 2. Java アプレット

## (1) Java プログラムリスト

[illegible]

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;
```

```

public class Nezumi_inu2 extends Applet {

    Button[] myBtn;                //ボタン型として宣言する
    Panel myPanel;                //パネル型として宣言する
    int imgNo,u=6,v=10;           //ねずみの速度 u、犬の速度 v
    //単位時間 h、経過時間 s、ねずみの x 座標 ux、ねずみの y 座標 uy、犬の x 座標 kx、犬の y 座標 ky
    double h=0.1,s=0,ux=0,uy=250,kx=0,ky=0;
    Image[] img;                 //イメージ型として宣言する
    MediaTracker myMt;           //メディアトラッカー型として宣言する
    int k;                        //奇跡カウンター
    double[] x1=new double[80];
    double[] y1=new double[80];
    double[] x2=new double[80];
    double[] y2=new double[80];
    int ct;                      //ループカウンター
    int x,y;                     //x 軸、y 軸描写に使用

//public void init() *****
    public void init() {

        setBackground(Color.lightGray);
        int count;               //ループカウンター
        imgNo=0;
        img=new Image[4];        //イメージの実体化

        img[0]=getImage(getDocumentBase(),"nezumi.jpg"); //ねずみ画像の読み込み
        img[1]=getImage(getDocumentBase(),"inu.jpg");    //犬画像の読み込み
        img[2]=getImage(getDocumentBase(),"tenr.jpg");   //赤い点画像の読み込み
        img[3]=getImage(getDocumentBase(),"tenb.jpg");   //青い点画像の読み込み

        myMt=new MediaTracker(this); //メディアトラッカーの実体化

        for (count=0;count<=3;count++) {
            myMt.addImage(img[count],0); //メディアトラッカーに画像を貼り付ける
        }

        try {
            myMt.waitForID(0);
        }
        catch (InterruptedException e) {
        }

        //setBackground(Color.green); //背景色の設定

        myBtn=new Button[2]; //ボタンの実体化
        myBtn[0]=new Button("初期化");
        myBtn[1]=new Button("次へ");

        myPanel=new Panel(); //パネルの実体化
        myPanel.setLayout(new GridLayout(1,2));
        for (count=0;count<=1;count++) {
            myPanel.add(myBtn[count]); //パネルにボタンを貼り付ける
        }

        setLayout(new BorderLayout()); //全体をボーダーレイアウトに設定する
        add("South",myPanel); // パネルを南に貼り付ける
    }
}

```

```

myBtn[1].addActionListener(new ActionListener() { //次へボタンを押したとき
    public void actionPerformed(ActionEvent e) {
        k=k+1;
        if (s!=0) {
            if (kx<ux) {
                kx=kx+5*((ux-kx)*v*h)/Math.sqrt((ux-kx)*
                    (ux-kx)+(uy-ky)*(uy-ky));
                ky=ky+5*((uy-ky)*v*h)/Math.sqrt((ux-kx)*
                    (ux-kx)+(uy-ky)*(uy-ky));
                x1[k]=kx;
                y1[k]=ky;

                ux=ux+5*u*h;
                uy=uy;
                x2[k]=ux;
                y2[k]=uy;

                s=s+h;

                repaint();
            }
        }
        else if (s==0) {
            kx=kx+5*((ux-kx)*v*h)/Math.sqrt((ux-kx)*
                (ux-kx)+(uy-ky)*(uy-ky));
            ky=ky+5*((uy-ky)*v*h)/Math.sqrt((ux-kx)*
                (ux-kx)+(uy-ky)*(uy-ky));
            x1[k]=kx;
            y1[k]=ky;

            ux=ux+5*u*h;
            uy=uy;
            x2[k]=ux;
            y2[k]=uy;

            s=s+h;

            repaint();
        }
    }
});

myBtn[0].addActionListener(new ActionListener() { //初期化ボタンを押したとき
    public void actionPerformed(ActionEvent e) {
        ux=0;uy=250;kx=0;ky=0;s=0;k=0;
        repaint();
    }
});

} //public void init()

//public void paint(Graphics g)*****
public void paint(Graphics g) {
    if (myMt.checkID(0)) {
        g.drawLine(2,0,2,250); // y 軸
        g.drawString(" y ",6+3,10+3);
        for (y=0;y<=250;y=y+50) {
            g.drawLine(0,y,4,y); // y 軸メモリ
        }
        g.drawString("40",6,50+5); // y 軸メモリ値
        g.drawString("30",6,100+5);
        g.drawString("20",6,150+5);
        g.drawString("10",6,200+5);
    }
}

```

```

g.drawLine(2,250,290,250);g.drawLine(290-5,250-5,290,250);g.drawLine
(290-5,250+5,290,250);// x 軸
g.drawString(" x ",285,240);
for (x=50;x<=250;x=x+50) {
    g.drawLine(x,248,x,252);// x 軸メモリ
}
g.drawString("0",6,247);// x 軸メモリ値
g.drawString("10",50-5,247);
g.drawString("20",100-5,247);
g.drawString("30",150-5,247);
g.drawString("40",200-5,247);
g.drawString("50",250-5,247);

g.drawImage(img[2],0,0,this);
g.drawImage(img[3],0,250,this);

for (ct=1;ct<=k;ct++) {
    g.drawImage(img[2],(int) x1[ct],(int) y1[ct],this);
    g.drawImage(img[3],(int) x2[ct],(int) y2[ct],this);
}
g.drawImage(img[0],(int) ux,(int) uy,this);
g.drawImage(img[1],(int) kx,(int) ky,this);

g.drawString("Copyright (C) K.Niwa 2001.11",80,325);
Font f1=new Font((g.getFont()).getName(),Font.PLAIN,16);
g.setFont(f1);
g.drawString("犬の位置   x =" + (int) kx/5,85,30);
g.drawString("           y =" + (int) (50-ky/5),160,30);
g.drawString("ねずみの位置 x =" + (int) ux/5,50,60);
g.drawString("           y =" + (int) (50-uy/5),160,60);
Font f2=new Font((g.getFont()).getName(),Font.BOLD,20);
g.setFont(f2);
g.drawString("経過時間(秒) =" + (float) s,70,100);

if (kx>=ux && s!=0) {
    g.drawString("追いついた!!",170,305);
}
}
} //public void paint(Graphics g)
} //public class Nezumi_inu2 extends Applet

```

## (2) HTML リスト

```

<HTML>
<HEAD>
< !-----
                                「ねずみを追う犬」
                                Copyright (C) K.Niwa 2001.10.14
                                ----->
</HEAD>
<BODY>
    <CENTER>
        <B>「ねずみを追う犬」</B>
        <BR><BR>
        <APPLET CODE="Nezumi_inu2.class" WIDTH="300" HEIGHT="350"></APPLET>
        <BR><BR>
        ※   犬(10 m/s の速度)は、ねずみ(6 m/s の速度)を追いかけます...
    </CENTER>
</BODY>
</HTML>

```

```

////////////////////////////////////
//
//                               「ねずみを追う犬」
//                               Copyright (C) K.Niwa 2002.08.24
//                               ( J a v a アプリケーション )
//
////////////////////////////////////

```

- 5 -

```

myBtn=new Button[2]; //ボタンの実体化
myBtn[0]=new Button("初期化");
myBtn[1]=new Button("次へ");

myPanel=new Panel(); //パネルの実体化
myPanel.setLayout(new GridLayout(1,2));
for (count=0;count<=1;count++) {
    myPanel.add(myBtn[count]); //パネルにボタンを貼り付ける
}

setLayout(new BorderLayout()); //全体をボーダーレイアウトに設定する
add("South",myPanel); // パネルを南に貼り付ける

myBtn[1].addActionListener(new ActionListener() { //次へボタンを押したとき
    public void actionPerformed(ActionEvent e) {
        k=k+1;
        if (s!=0) {
            if(kx<ux) {
                kx=kx+5*((ux-kx)*v*h)/Math.sqrt((ux-kx)*
                    (ux-kx)+(uy-ky)*(uy-ky));
                ky=ky+5*((uy-ky)*v*h)/Math.sqrt((ux-kx)*
                    (ux-kx)+(uy-ky)*(uy-ky));
                x1[k]=kx;
                y1[k]=ky;

                ux=ux+5*u*h;
                uy=uy;
                x2[k]=ux;
                y2[k]=uy;

                s=s+h;

                repaint(); //repaint()メソッドを呼び出す
            }
        }
        else if (s==0) {
            kx=kx+5*((ux-kx)*v*h)/Math.sqrt((ux-kx)*
                (ux-kx)+(uy-ky)*(uy-ky));
            ky=ky+5*((uy-ky)*v*h)/Math.sqrt((ux-kx)*
                (ux-kx)+(uy-ky)*(uy-ky));
            x1[k]=kx;
            y1[k]=ky;

            ux=ux+5*u*h;
            uy=uy;
            x2[k]=ux;
            y2[k]=uy;

            s=s+h;

            repaint(); //repaint()メソッドを呼び出す
        }
    }
});

myBtn[0].addActionListener(new ActionListener() { //初期化ボタンを押したとき
    public void actionPerformed(ActionEvent e) {
        ux=0;uy=250+20+20;kx=0;ky=0+20+20;s=0;k=0;
        repaint();
    }
});

} //public FNezumi_inu2()

```

```

//public void paint(Graphics g)*****
    public void paint(Graphics g) {
        if (myMt.checkID(0)) {
            g.drawLine(2+2,0+20+20,2+2,250+20+20); // y 軸
            g.drawString(" y ",6+3+2,10+3+20+20);
            for (y=0+20+20;y<=250+20+20;y=y+50) {
                g.drawLine(0+2,y,4+2,y); // y 軸メモリ
            }
            g.drawString("40",6+2,50+5+20+20); // y 軸メモリ値
            g.drawString("30",6+2,100+5+20+20);
            g.drawString("20",6+2,150+5+20+20);
            g.drawString("10",6+2,200+5+20+20);

            g.drawLine(2,250+20+20,290,250+20+20); // x 軸
            g.drawLine(290-5,250-5+20+20,290,250+20+20);
            g.drawLine(290-5,250+5+20+20,290,250+20+20);
            g.drawString(" x ",285,240+20+20);
            for (x=50;x<=250;x=x+50) {
                g.drawLine(x,248+20+20,x,252+20+20); // x 軸メモリ
            }
            g.drawString("0",6,247+20+20); // x 軸メモリ値
            g.drawString("10",50-5,247+20+20);
            g.drawString("20",100-5,247+20+20);
            g.drawString("30",150-5,247+20+20);
            g.drawString("40",200-5,247+20+20);
            g.drawString("50",250-5,247+20+20);

            g.drawImage(img[2],0,0+20+20,this);
            g.drawImage(img[3],0,250+20+20,this);
            for (ct=1;ct<=k;ct++) {
                g.drawImage(img[2],(int) x1[ct],(int) y1[ct],this);
                g.drawImage(img[3],(int) x2[ct],(int) y2[ct],this);
            }
            g.drawImage(img[0],(int) ux,(int) uy,this);
            g.drawImage(img[1],(int) kx,(int) ky,this);

            g.drawString("Copyright (C) K.Niwa 2002.08",80,325+20+20);
            Font f1=new Font((g.getFont()).getName(),Font.PLAIN,16);
            g.setFont(f1);
            g.drawString("犬の位置 x =",(int) kx/5,85,30+20+20);
            g.drawString(" y =",(int) (50-ky/5+8),160,30+20+20);
            g.drawString("ねずみの位置 x =",(int) ux/5,50,60+20+20);
            g.drawString(" y =",(int) (50-uy/5+8),160,60+20+20);
            Font f2=new Font((g.getFont()).getName(),Font.BOLD,20);
            g.setFont(f2);
            g.drawString("経過時間(秒) =",(float) s,70,100+20+20);

            if (kx>=ux && s!=0) {
                g.drawString("追いついた!!",170,305+20+20);
            }
        }
    } //public void paint(Graphics g)

/***** public static void main メソッド *****/
    public static void main(String[] args) {
        Frame w=new FNezumi_inu2();
        w.show();
    } //public static void main(String[] args)

} //public class FNezumi_inu2 extends Frame

```