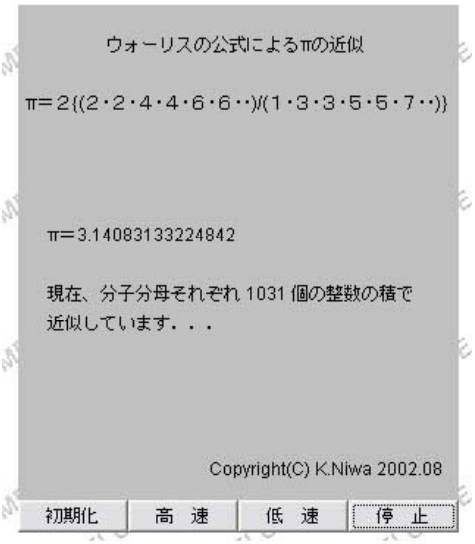
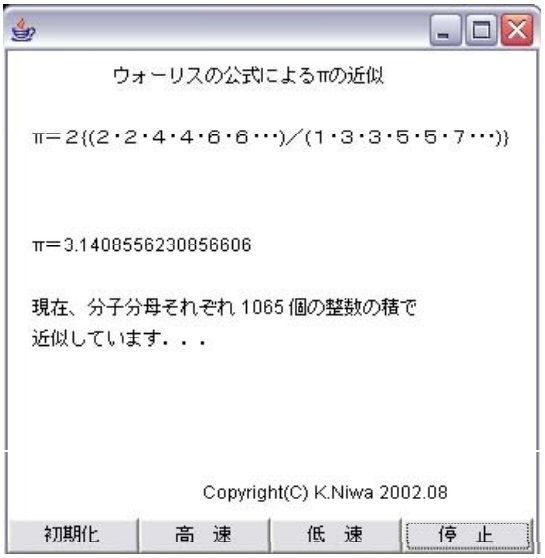
【ウォーリスの公式】



［Ｊａｖａアプレット］



［Ｊａｖａアプリケーション］

１．はじめに

次のウォーリスの公式を用いてπの近似値を求めてみましょう。

［ウォーリスの公式］

$$\pi = 2\ \{\frac{2\cdot 2\cdot 4\cdot 4\cdot 6\cdot 6\cdot\cdot\cdot}{1\cdot 3\cdot 3\cdot 5\cdot 5\cdot 7\cdot\cdot\cdot}\}$$

シミュレーションソフト「ウォーリスの公式によるπの近似」を使って、πの近似値が求まる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

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                                                                                        //
//                                          「ウォーリスの公式によるπの近似」                                                             //
//                                          Copyright（C）K.Niwa  2002.08.11                                                              //
//                                              （Ｊａｖａアプレット）                                                                    //
//                                                                                                                                        //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//クラスの読み込み
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

/***** public class Wourisu extends Applet implements Runnable ************************************/
public class   Wourisu extends Applet implements Runnable{          //スレッドを使えるようにする

//変数とオブジェクトの型宣言
        Thread myTh;                        //スレッド型で宣言する
        Button[] myBtn;                     //ボタン型配列で宣言する
        Panel myPanel;                      //パネル型で宣言する
```

```
    int flag=0;
    int Speed=200;                          //速度
    int ct=0;                               //実験回数
    int count;                              //ループカウンター
    double pai;                             //πの近似値
    double sa=1;                            //πの近似値を求める過程で使用

/*****  public  void  init()ﾒｿｯﾄﾞ**********************************************************************/
    public void init(){
            setBackground(Color.lightGray);             //背景色をグレーにする
            myTh=null;                                  //スレッドの初期化

            myBtn=new Button[4];                        //ボタンの実体化
            myBtn[0]=new Button("初期化");
            myBtn[1]=new Button("高　速");
            myBtn[2]=new Button("低　速");
            myBtn[3]=new Button("停　止");

            myPanel=new Panel();                        //パネルの実体化
            myPanel.setLayout(new GridLayout(1,4));
            for (count=0;count<=3;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する
            add("South",myPanel);                       //パネルを南に貼り付ける

            myBtn[0].addActionListener(new ActionListener(){    //初期化ボタンの定義
                    public void actionPerformed(ActionEvent e){
                            flag=0;             //ボタンの識別子
                            repaint();
                    }
            });

            myBtn[1].addActionListener(new ActionListener(){    //高速ボタンの定義
                    public void actionPerformed(ActionEvent e){
                            flag=1;             //ボタンの識別子
                            Speed=20;                                   //速度
                            repaint();
                    }
            });

            myBtn[2].addActionListener(new ActionListener(){    //低速ボタンの定義
                    public void actionPerformed(ActionEvent e){
                            flag=2;             //ボタンの識別子
                            Speed=200;                                  //速度
                            repaint();
                    }
            });

            myBtn[3].addActionListener(new ActionListener(){    //停止ボタンの定義
                    public void actionPerformed(ActionEvent e){
                            flag=3;             //ボタンの識別子
                            repaint();
                    }
            });

    }//public void init()

/*****  public  void  start()ﾒｿｯﾄﾞ**********************************************************************/
    public void start(){
            if (myTh==null){
                    myTh=new Thread(this);              //スレッドの実体化
                    myTh.start();                       //スレッドの開始
            }
```

```
	}

/******  public void run()ﾒｿｯﾄﾞ **********イベントなしで動作する*****************************/
	public void run(){
		while (true){
			try{
				myTh.sleep(Speed);
			}
			catch (InterruptedException e) {}
			if (flag==1 || flag==2){
				repaint();
			}
		}
	}

/*****  public void paint(Graphics g)ﾒｿｯﾄﾞ ************************************************/
	public void paint(Graphics g){

	//初期状態または初期化ボタンを押したときのイベント処理
		if(flag==0){

			//g.clearRect(0,0,300,360);		//全体のクリア
			ct=0;					//第何項目までの和であるかの初期化
			sa=1;					//πの近似値を求める過程で使用
			pai=0;					//πの近似値の初期化

			g.drawString("ウォーリスの公式によるπの近似",70-10,20+10);
			g.drawString("π＝２{(２・２・４・４・６・６‥)/(１・３・３・５・５・７‥)}"
									,20-15,60+10);
			g.drawString("π＝",30-10,160);
			g.drawString("現在、分子分母それぞれ "+" "+" 個の整数の積で"
					,30-10,300-100); //第何項目までの和であるかを表示
		//第何項目までの和であるかを表示
			g.drawString("近似しています... ",30-10,300-100+20);
			g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);	//作者表示

		}//if(flag==0)

	//高速ボタンまたは低速ボタンを押したときのイベント処理
		else if (flag==1 || flag==2){
			if (ct<2147483647){
				ct=ct+1;
			}
			else{
				flag=3;
			}

			sa=sa*(((double)2*ct*2*ct)/((double)(2*ct-1)*(2*ct+1)));
			pai=(double)2*sa;

			g.drawString("ウォーリスの公式によるπの近似",70-10,20+10);
			g.drawString("π＝２{(２・２・４・４・６・６‥)/(１・３・３・５・５・７‥)}"
									,20-15,60+10);
			g.drawString("π＝"+pai,30-10,160);
			g.drawString("現在、分子分母それぞれ "+ct+" 個の整数の積で"
					,30-10,300-100); //第何項目までの和であるかを表示
		//第何項目までの和であるかを表示
			g.drawString("近似しています... ",30-10,300-100+20);

			g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);	//作 者 表 示

		}//else if (flag==1 || flag==2)
```

```
        //停止ボタンを押したときのイベント処理
                if(flag==3){
                        g.drawString("ウォーリスの公式によるπの近似",70-10,20+10);
                        g.drawString("π＝２{(２・２・４・４・６・６‥)/(１・３・３・５・５・７‥)}"
                                                                        ,20-15,60+10);
                        g.drawString("π＝"+pai,30-10,160);
                        g.drawString("現在、分子分母それぞれ "+ct+" 個の整数の積で"
                                        ,30-10,300-100); //第何項目までの和であるかを表示
        //第何項目までの和であるかを表示
                        g.drawString("近似しています... ",30-10,300-100+20);
                        g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);          //作 者 表 示
                }//if(flag==3)

        }//public void paint(Graphics g)

}//public class Gregory extends Applet implements Runnable
```

（2）ＨＴＭＬリスト

```
<HTML>
        <HEAD>
< !-------------------------------------------------------------------------------------------------------------------------------
                        「ウォーリスの公式によるπの近似」
                          Copyright（C）K.Niwa 2002.08.11
------------------------------------------------------------------------------------------------------------------------------->
        </HEAD>
        <BODY>
                <CENTER>
                        <B>「ウォーリスの公式によるπの近似」</B>
                        <BR><BR>
                        <APPLET CODE="Wourisu.class" WIDTH="300" HEIGHT="360"></APPLET>
                </CENTER>
        </BODY>
</HTML>
```

３．Ｊａｖａアプリケーション・プログラムリスト

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                                                                              //
//                                「Ｆウォーリスの公式によるπの近似」                                                          //
//                                    Copyright（C）K.Niwa 2002.08.17                                                           //
//                                                                                                                              //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

// クラスの読み込み
import java.awt.*;
import java.awt.event.*;
import java.lang.Math;

/****** public class FWourisu extends Frame implements Runnable **********************************/
public class   FWourisu extends Frame implements Runnable{        //スレッドを使えるようにする

//変数とオブジェクトの型宣言
        Thread myTh;                            //スレッド型で宣言する
        Button[] myBtn;                         //ボタン型配列で宣言する
        Panel myPanel;                          //パネル型で宣言する
        int flag=0;
        int Speed=200;                          //速度
        int ct=0;                               //実験回数
        int count;                              //ループカウンター
        double pai;                             //πの近似値
        double sa=1;                            //πの近似値を求める過程で使用
```

```
//*****フレームとイベント処理の定義**********************************************************
	public FWourisu(){

		setSize(330+20,360);                                    //フレームの大きさ
		addWindowListener(new WindowAdapter(){                  //閉じるボタンイベント処理
			public void windowClosing(WindowEvent e){
				System.exit(0);
			}
		});

		myTh=null;                                              //スレッドの初期化
		if (myTh==null){
			myTh=new Thread(this);                          //スレッドの実体化
			myTh.start();                                   //スレッドの開始
		}

		myBtn=new Button[4];                                    //ボタンの実体化
		myBtn[0]=new Button("初期化");
		myBtn[1]=new Button("高　速");
		myBtn[2]=new Button("低　速");
		myBtn[3]=new Button("停　止");

		myPanel=new Panel();                                    //パネルの実体化
		myPanel.setLayout(new GridLayout(1,4));                 //ﾊﾟﾈﾙをｸﾞﾘｯﾄﾞﾚｲｱｳﾄにする
		for (count=0;count<=3;count++){
			myPanel.add(myBtn[count]);                      //パネルにボタンを貼り付ける
		}
		setLayout(new BorderLayout());                          //全体をﾎﾞｰﾀﾞｰﾚｲｱｳﾄにする
		add("South",myPanel);                                   //パネルを南に貼り付ける

		myBtn[0].addActionListener(new ActionListener(){        //初期化ボタンの定義
			public void actionPerformed(ActionEvent e){
				flag=0;                                 //ボタン識別子
				repaint();
			}
		});

		myBtn[1].addActionListener(new ActionListener(){        //高速ボタンの定義
			public void actionPerformed(ActionEvent e){
				flag=1;                                 //ボタン識別子
				Speed=20;                                       //速度
				repaint();
			}
		});

		myBtn[2].addActionListener(new ActionListener(){        //低速ボタンの定義
			public void actionPerformed(ActionEvent e){
				flag=2;                                 //ボタン識別子
				Speed=200;                                      //速度
				repaint();
			}
		});

		myBtn[3].addActionListener(new ActionListener(){        //停止ボタンの定義
			public void actionPerformed(ActionEvent e){
				flag=3;                                 //ボタン識別子
				repaint();
			}
		});

	}//public FWourisu()
```

```
/*****  public  void  run()ﾒｿｯﾄﾞ  ************************************************************
        public void run(){
                while (true){
                        try{
                                myTh.sleep(Speed);
                        }
                        catch (InterruptedException e) {}
                        if (flag==1 || flag==2){
                                repaint();
                        }
                }
        }

/******  public void paint(Graphics g)ﾒｿｯﾄﾞ  ************************************************/
        public void paint(Graphics g){
        //初期状態または初期化ボタンを押したときのイベント処理

                if(flag==0){
                        //g.clearRect(0,0,350,360);             //全体のクリア
                        ct=0;                                   //第何項目までの和であるかの初期化
                        sa=1;                                   //πの近似値を求める過程で使用
                        pai=0;                                  //πの近似値の初期化
                        g.drawString("　ウォーリスの公式によるπの近似",70-10,20+10+20);
                        g.drawString("π＝２{(２・２・４・４・６・６･･･)／(１・３・３・５・５・７･･･)}"
                                                                        ,20,60+10+20);
                        g.drawString("π＝",30-10,160);
                        g.drawString("現在、分子分母それぞれ "+" "+" 個の整数の積で"
                                        ,30-10,300-100); //第何項目までの和であるかを表示
                //第何項目までの和であるかを表示
                        g.drawString("近似しています... ",30-10,300-100+20);
                        g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);       //作者表示
                }//if(flag==0)

        //高速ボタンまたは低速ボタンを押したときのイベント処理
                else if (flag==1 || flag==2){
                        if (ct<2147483647){
                                ct=ct+1;
                        }
                        else{
                                flag=3;
                        }
                        sa=sa*(((double)2*ct*2*ct)/((double)(2*ct-1)*(2*ct+1)));
                        pai=(double)2*sa;
                        g.drawString("　ウォーリスの公式によるπの近似",70-10,20+10+20);
                        g.drawString("π＝２{(２・２・４・４・６・６･･･)／(１・３・３・５・５・７･･･)}"
                                                                        ,20,60+10+20);
                        g.drawString("π＝"+pai,30-10,160);
                        g.drawString("現在、分子分母それぞれ "+ct+" 個の整数の積で"
                                        ,30-10,300-100); //第何項目までの和であるかを表示
                //第何項目までの和であるかを表示
                        g.drawString("近似しています... ",30-10,300-100+20);
                        g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);         //作 者 表 示
                }//else if (flag==1 || flag==2)

        //停止ボタンを押したときのイベント処理
                if(flag==3){
                        g.drawString("　ウォーリスの公式によるπの近似",70-10,20+10+20);
                        g.drawString("π＝２{(２・２・４・４・６・６･･･)／(１・３・３・５・５・７･･･)}"
                                                                        ,20,60+10+20);
                        g.drawString("π＝"+pai,30-10,160);
                        g.drawString("現在、分子分母それぞれ "+ct+" 個の整数の積で"
                                        ,30-10,300-100); //第何項目までの和であるかを表示
                //第何項目までの和であるかを表示
                        g.drawString("近似しています... ",30-10,300-100+20);
```

```
                        g.drawString("Copyright(C) K.Niwa 2002.08",130,330-10);       //作者表示
                }//if(flag==3)
        }//public void paint(Graphics g)

//*****main メソッドでＪａｖａアプリケーションには必ず必要である**************************
        public static void main(String[] args){
                Frame w=new FWourisu();
                w.show();
        }//public static void main(String[] args)

}//public class FWourisu extends Frame implements Runnable
```