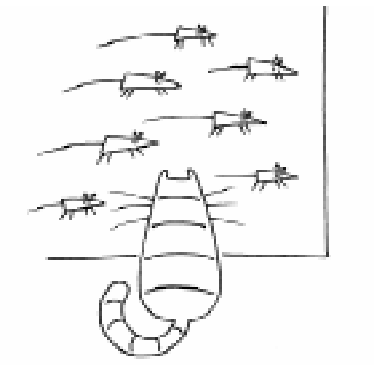
Дмитриева Марина Валерьевна

ОСНОВЫ ПРОГРАММИРОВАНИЯ ДЛЯ ИНТЕРНЕТ. АППЛЕТЫ. ЗАНЯТИЕ 3. ИЗОБРАЖЕНИЯ И ПРОСТЫЕ АНИМАЦИОННЫЕ ЭФФЕКТЫ

В статье рассматриваются методы, позволяющие вставлять изображения в аплет. На основе потока приводятся программы, реализующие анимационные эффекты.

Изображения в интернет-документах могут использоваться с различной целью. Так же, как и в печатных изданиях, изображения могут иллюстрировать теоретический и любой другой текст, помогая передавать суть документа. С помощью изображений представляются такие данные, как схемы движения, расположение объектов и т. п. Изображения добавляются на страницы для придания им привлекательного вида, являются основой многих визуальных эффектов. Графические файлы используются при разработке компьютерных игр и во многих других приложениях.



Изображения добавляются на страницы для придания им привлекательного вида...

Изображение (рисунок) можно вставить непосредственно в HTML-документ с помощью тега **<img>** или отобразить в составе апплета.

Изображение является одним из ресурсов апплета, расположенном в файле с определенным URL-адресом. Это файл в графическом формате GIF или JPEG. Изображение в Java – это объект класса **Image**, представляющий прямоугольный массив пикселей.

Язык Java имеет готовые классы для обработки изображений. Для загрузки изображения используется метод **getImage**, которому необходимо указать полный путь к графическому файлу, в котором хранится изображение.

Рисование изображения может потребовать значительного времени для скачивания его с указанного URL-адреса и преобразования к формату, который может быть выведен на устройство. Изображение на экране воспроизводится с помощью логических методов **drawImage** класса **Graphics**.

В следующих двух примерах будут представлены программы, позволяющие вставлять изображения в аплет, подробно описаны методы **getImage** и **drawImage**.

ВСТАВКА ИЗОБРАЖЕНИЯ В АПЛЕТ

Создадим апплет для вывода изображения (листинг 1). В программе описывается переменная **pict** класса **Image**. Для загрузки изображения используется метод **getImage**, которому, как упоминалось ранее, необходимо указать полный путь к файлу. Путь формируется с помощью двух параметров. При помощи метода **getCodeBase()** определяется URL-адрес кода апплета. Второй параметр – строка, задающая путь к файлу с изображением относительно базового URL, заданного в первом параметре. Изображение на экране воспроизводится с помощью логического метода **drawImage**:

```
drawImage(Image img, int x,
 int y, int width, int height,
 ImageObserver obs)
```

Аргументы **(x,y)** задают координаты левого верхнего угла изображения **img**, аргументы **width** и **height** – ширину и высоту изображения на экране, аргумент **obs** – ссылку на объект, следящий за процессом загрузки изображения. Последнему аргументу можно задать значение **this**.

Размеры изображения **pict** можно получить с помощью методов **getWidth()**, **getHeight()** класса **Image**. При воспроизведении изображения в приведенной программе размер выводимого изображения уменьшен в два раза. В строке статуса выводится значение, выдаваемое методом **getCodeBase()**, определяющее URL-адрес кода апплета. Координаты левого верхнего угла выводимого изображения: **(20,20)**.

Рис. 1. Изображение в апплете

На рис. 1 представлен апплет после его запуска. Обратите внимание на строку статуса.

ВСТАВКА ИЗОБРАЖЕНИЯ ПО ЦЕНТРУ

Напишем программу, которая помещает изображение в центр области. Для загрузки изображения воспользуемся методом getImage, первый параметр которого – URL директории, содержащий HTML-файл, второй – строка, задающая путь к

Листинг 1. Вставка изображения в апплет

```
import java.applet.Applet;
import java.awt.*;
public class SloadImApp1  extends Applet {
    Image pict;
    public void init() {
     resize(300,300);
       pict = getImage(getCodeBase(), "../img/img1.jpg");
     }
    public void paint(Graphics g) {
        int w = pict.getWidth(this)/2;
        int h = pict.getHeight(this)/2;
        g.drawImage(pict, 20, 20, w, h, this);
        showStatus(""+getCodeBase());
     }
}
```

Рис. 2. Изображение, расположенное по центру

файлу с изображением относительно базового URL, заданного в первом параметре.

Если воспроизводить изображение с его реальными размерами, то можно воспользоваться упрощенной версией метода **drawImage**, при вызове которого задается изображение, координаты левого верхнего угла области и ссылка на объект, следящий за процессом загрузки изображения:

```
drawImage(Image img, int x, int y,
                    ImageObserver obs)
```

На практике эта версия метода выводит изображение быстрее, поскольку не требуется вычислять размеры выводимого изображения. В программе определяются координаты верхнего левого угла изображения таким образом, чтобы изображение размещалось по центру. Если менять размеры окна апплета, то изображение будет располагаться по центру и в окне с измененными размерами.

В листинге 2 приводится текст программы, в которой загружается изображение, в строке статуса отображается значение метода **getDocumentBase()**.

На рис. 2 представлен апплет, в котором загружаемое изображение размещается по центру. В строке статуса представлена часть URL-адреса, полученная с помощью метода **getDocumentBase()**.

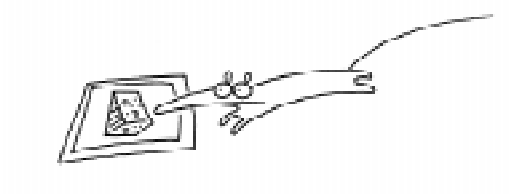
Для воспроизведения изображения в окне апплета необходимо указать, «кто» будет рисовать (объект класса **Graphics**), что рисовать (объект **Image**), где рисовать (координаты **x** и **y**) и кто отвечает за то, как рисуется изображение (**ImageObserver**).

ПЕРЕМЕЩЕНИЕ КАРТИНКИ С ПОМОЩЬЮ КУРСОРА МЫШИ

Напишем программу, которая позволяет перемещать картинку с помощью мыши. Если курсор мыши попадает в область изображения, то рисунок можно перетащить, нажав на клавишу мыши. В строке статуса задаются текущие координаты курсора мыши при перемещении

Листинг 2. Вставка изображения в апплет по центру

```
import java.applet.*;
import java.awt.*;
public class SloadImApp2  extends Applet {
    Image pict;
    int w,h,w1,h1;
    public void init() {
      pict  = getImage(getDocumentBase(), "../img/m1.gif");
     }
    public void paint(Graphics g) {
     w = getWidth();
     h = getHeight();
     w1 = pict.getWidth(this);
     h1 = pict.getHeight(this);
       g.drawImage(pict,w/2-w1/2, h/2-h1/2, this);
       showStatus(""+getDocumentBase());
     }
}
```



Напишем программу, которая позволяет перемещать картинку с помощью мыши.

курсора и при перемещении курсора с нажатой клавишей (перетаскивании).

Логическая переменная имеет значение **true**, если происходит перетаскивание рисунка. Если клавиша мыши отпущена, то рисунок перемещен на новое место, значение логической переменной изменится на **false**.

В программе используются два изображения. Одно из них играет роль фонового (облако). Оно растянуто на всю область апплета. Второе изображение (солнышко) можно перетаскивать с помощью курсора мыши. Когда изображение перетаскивается, вокруг него рисуется прямоугольная рамка. В программе для организации работы с мышью используются методы интерфейсов **MouseListener** и **MouseMotionListener**. В листинге 3 представлен текст программы.

На рис. 3 изображен момент, когда курсор мыши располагается в области

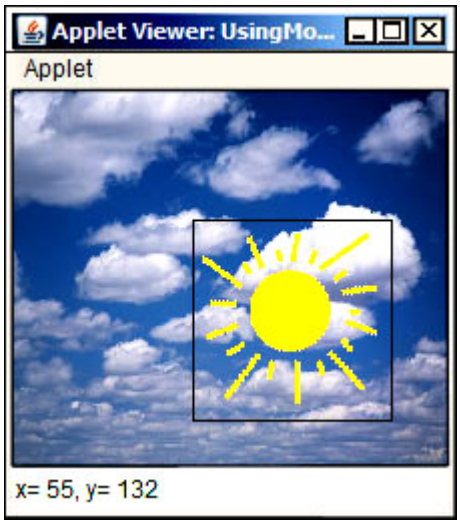


Рис. 3. Перемещение рисунка с нажатой клавише мыши

рисунка и клавиша мыши нажата, то есть рисунок можно перетаскивать.

ДВИЖЕНИЕ КАРТИНКИ ПО РАЗНЫМ НАПРАВЛЕНИЯМ

Напишем программу, которая позволяет перемещать рисунок по четырем направлениям: вверх, вниз, вправо, влево. В зависимости от того, в какую сторону должно быть движение, выбирается соответствующая картинка. Все изображения

Листинг 3. Перетаскивание изображения

```
import java.awt.Graphics;
import java.awt.Image;
import java.awt.event.*;
import java.applet.Applet;
// Перемещение изображения с помощью курсора мыши
public class UsingMouseActionsApplet extends Applet
         implements   MouseListener, MouseMotionListener {
    Image image, background;
// координаты изображения на экране
    int x = 50;
    int y = 50;
    int width = 100;
    int height = 100;
// текущие координаты курсора мыши
    int curX, curY;
// перемещение изображения
    boolean imageIsTaken;
    public void init() {
```

```
      image = getImage(getDocumentBase(), "../img/sun.gif");
      background = getImage(getDocumentBase(), "../img/oblako01.jpg");
      addMouseListener(this);
      addMouseMotionListener(this);
    }
// клавиша мыши нажата и удерживается
    public void mousePressed(MouseEvent e) {
// курсор мыши в области изображения
       if (e.getX() > x && e.getX() < x + width
           && e.getY() > y && e.getY() < y + height) {
           imageIsTaken = true;
           curX = e.getX();
           curY = e.getY();
           showStatus("Перетаскивание изображения");
           repaint();
         }
      }
// клавиша мыши отпущена
     public void mouseReleased(MouseEvent e) {
        imageIsTaken = false;
        repaint();
      }
// другие методы интерфейса MouseListener
      public void mouseClicked(MouseEvent e) {}
      public void mouseEntered(MouseEvent e) {}
      public void mouseExited(MouseEvent e) {}
// движение курсора мыши с нажатой клавишей
      public void mouseDragged(MouseEvent e) {
// движение картинки за курсором, если она "захвачена"
       if (imageIsTaken) {
            x = x + e.getX() - curX;
            y = y + e.getY() - curY;
            repaint();
          }
          mouseMoved(e);
       }
// перемещение курсора мыши
      public void mouseMoved(MouseEvent e) {
          curX = e.getX();
          curY = e.getY();
          showStatus("x= "+ curX + ", y= " + curY);
      }
      public void update(Graphics g) {
         paint(g);
      }
      public void paint(Graphics g) {
        g.drawImage(background, 0, 0, getWidth(), getHeight(), this);
        g.drawImage(image, x, y, width, height, this);
        if (imageIsTaken)  {
          // рамка вокруг картинки при перемещении
         g.drawRect(x, y, width, height);
        }
      }
}
```

объединяются в массив изображений. Управление движением будет производиться с помощью клавиш. Нажатая клавиша анализируется, и в зависимости от направления движения выбирается та картинка, которая соответствует движению вперед. Если при движении дошли до границы окна апплета, то выход за границу будет блокирован. Первоначально рисунок располагается в центре области.

В строке статуса выводится сообщение о том, в каком направлении сделан последний шаг. В листинге 4 представлен текст программы.

На рис. 4 зафиксирован последний шаг после перемещения изображения по разным направлениям.

Движение картинки по разным направлениям

ЧЕРЕДОВАНИЕ ИЗОБРАЖЕНИЙ

Напишем программу, которая позволяет загрузить одно изображение, а далее управлять загрузкой изображений с помощью клавиш мыши. При щелчке левой кнопкой мыши появляется следующее изображение, при щелчке правой – загружается предыдущее изображение. Загрузка изображений происходит «по кругу». Если загружено последнее изображение, и нажата левая кнопка (следующее), то следующим загруженным изображением станет первое. Если загружено первое изображение, и нажата правая кнопка мыши (предыдущее), то следующим будет загружено последнее изображение.

В программе формируется массив изображений. Предполагается, что файлы с изображениями хранятся в одной папке. Поэтому формирование массива можно осуществить с помощью цикла, в котором параметр цикла используется при формировании имени файла с изображением.

Переменная **`curPict`** содержит индекс загруженного изображения. В строке статуса выдается номер выводимого изображения.

Проверка, какая кнопка мыши нажата, осуществляется с помощью переключателя. Константа **`MouseEvent.BUTTON1`** соответствует левой кнопке мыши, константа **`MouseEvent.BUTTON3`** – правой кнопке. В листинге 5 представлен текст программы, осуществляющий замену изображения с помощью клавиш мыши.

Все изображения рисуются в центре области апплета. Так как все изображения имеют различные размеры, то при необходимости выводимое изображение уменьшается с сохранением пропорций исходного изображения. В программе вычисляется коэффициент сжатия изображения по ширине и по высоте. На рис. 5 приведен апплет с загруженным изображением. В строке статуса указан номер загруженного изображения.

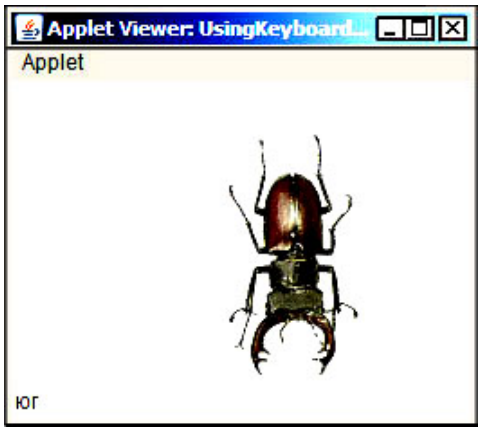


Рис. 4. Управление движением рисунка с помощью клавиш

Листинг 4. Движение по четырем направлениям

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class UsingKeyboardActionsApplet extends Applet
     implements   KeyListener{
// Используем эти парамаетры для движения изображения.
     int x = getWidth()/2;
     int y = getHeight()/2;
// размер изображения
    int d = 140;
// массив изображений (картинок)
    Image[] pict = new Image[4];
     int curPict = 0;
     String msg="";
// формирование масива изображений и установка блока прослушивания
     public void init() {
      pict[0] = getImage(getDocumentBase(),
                                    "../img/lucanida/lucanidaUp.gif");
      pict[1] = getImage(getDocumentBase(),
                                 "../img/lucanida/lucanidaRight.gif");
      pict[2] = getImage(getDocumentBase(),
                                  "../img/lucanida/lucanidaDown.gif");
      pict[3] = getImage(getDocumentBase(),
                                  "../img/lucanida/lucanidaLeft.gif");
        addKeyListener(this);
     }
// нажатие клавиши клавиатуры
     public void keyPressed(KeyEvent e) {
// от нажатой клавиши зависит, в какую сторону двигаться
        switch ((char)e.getKeyCode()) {
           case "W": x -= 15; curPict = 3; msg="запад";  break;
           case "E": x += 15; curPict = 1; msg="восток"; break;
           case "N": y -= 15; curPict = 0; msg="север"; break;
           case "S": y += 15; curPict = 2; msg="юг"; break;
         }
// граница области окна апплета
          if (x > getWidth() - d) x = getWidth() - d;
          else
            if (x < 0) x = 0;
          if (y > getHeight() - d) y = getHeight() - d;
          else
            if (y < 0) y = 0;
          showStatus(msg);
          repaint();
     }
// другие методы интерфейса KeyListener
     public void keyReleased (KeyEvent evt) {}
     public void keyTyped(KeyEvent evt) {}
// рисование
    public void paint(Graphics g) {
      int w = pict[curPict].getWidth(this);
     int h = pict[curPict].getHeight(this);
      g.drawImage(pict[curPict], x, y, w,h,this);
     }
}
```

Листинг 5. Чередование изображений с помощью клавиш мыши

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class PreLoadImageApplet1 extends Applet
    implements   MouseListener{
// число изображений
     int pictCount = 9;
// массив изображений
    Image[] picts = new Image[pictCount];
    int curPict = 0;
// переключение между изображениями по кнопкам мыши
     public void mouseClicked(MouseEvent e) {
// проверка, какая кнопка мыши нажата
        switch (e.getButton()) {
           case MouseEvent.BUTTON1: // левая кнопка мыши
              curPict++;
              if (curPict >= pictCount) {curPict = 0;} break;
           case MouseEvent.BUTTON3: // правая кнопка мыши
               curPict—;
               if (curPict < 0) {curPict = pictCount - 1;}break;
           default: curPict = 0;
        }
        int k = curPict+1;
        showStatus("изображение " + k);
        repaint();
   }
     public void init() {
      addMouseListener(this);
      for (int i = 0; i < pictCount; i++) {
         picts[i] = getImage(getDocumentBase(),
                             "../img/aivasovskiy/pic" + i + ".jpg");
      }
        repaint();
      }
// рисуем выбранное изображение с сохранением пропорций в центре
апплета
  public void paint(Graphics g) {
   double coeff1 = (double)picts[curPict].getWidth(this) / getWidth();
   double coeff2 = (double)picts[curPict].getHeight(this) / getHeight();
   if (coeff1 > coeff2) {
    int height =
           (int)Math.round(picts[curPict].getHeight(this) / coeff1);
     g.drawImage(picts[curPict],0,
                 (getHeight() - height)/2,getWidth(),height,this);
    } else {
     int width = (int)Math.round(picts[curPict].getWidth(this) / coeff2);
      g.drawImage(picts[curPict],
                  (getWidth()-width)/2,0,width,getHeight(),this);
     }
  } // paint
// другие методы интерфейса MouseListener
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
  }
```

Рис. 5. Чередование изображений с помощью клавиш мыши

ПОТОКИ

Поток (**thread**) – параллельно выполняемая часть («нить») программы. Язык Java имеет поддержку многопоточного программирования на уровне языковых конструкций и встроенных библиотек. Новый поток в Java является объектом, имеющим конструктор и метод запуска. Каждый поток в Java характеризуется именем и приоритетом. Приоритет определяет очередность выполнения потоков при наличии нескольких потоков, готовых к выполнению.

Новый поток может быть создан двумя способами. В первом случае можно создать объект нового класса, являющегося потомком класса **Thread**, содержащего все необходимые средства для управления потоком.

Во втором случае для создания потока можно определить объект нового класса, реализующий интерфейс **Runnable**, имеющий метод **run()**, который и выполняет работу, возложенную на поток.

Метод **start()** сообщает потоку о том, что надо запустить метод **run()**. Для приостановки текущего потока на указанный промежуток времени используется метод **Thread.sleep(t)**, где **t** – время в миллисекундах, на которое поток «засыпает». Метод **sleep()** может завершиться исключительной ситуацией **InterruptedException**, переводящей поток в состояние ожидания. Далее состояние ожидания может быть прервано, то есть «заснувший» поток может быть вновь «разбужен».

ИСКЛЮЧЕНИЯ

В языке Java большинство ошибок вызывают так называемые исключительные ситуации или исключения. Если некоторый метод вызывает исключительную ситуацию, то программа может попытаться выполнить метод, а в случае возникновения исключительной ситуации попытаться перехватить (**catch**) и обработать ее. Исключения бывают контролируемые и неконтролируемые. Контролируемые исключения – потомки класса **Exception** могут быть обработаны пользователем. Рассмотрим схему в листинге 6.

Код, который может вызвать исключения, помещается в программный блок **try**, а код обработки этого исключения располагается в блоке **catch**. Если какой-то метод в блоке **try** возбуждает исключение, то игнорируются остальные операторы в блоке **try**, и происходит переход на блок **catch**, в котором программа обрабатывает исключения. Если все проис-

Листинг 6. Обработка исключений

```
try
{…}
catch ()
{…}
```

ходит нормально, то весь код внутри блока **try** выполняется, а блок **catch** пропускается.

Программный блок **catch** перехватывает исключение, возбужденное системой Java. После слова **catch** можно записать объект-исключение, заключенный в круглые скобки.

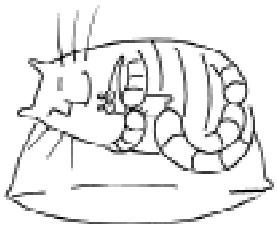
АНИМАЦИОННЫЕ ЭФФЕКТЫ. СЛАЙД-ШОУ

Напишем программу, создающую анимационный эффект, основанный на смене изображений. Файлы с изображениями хранятся в одной папке и имеют имена **ball0, ball1, ..., ball9**.

В программе с помощью цикла формируется массив изображений **pict**. Эффект анимации будет достигаться использованием потока **runner**, который описывается и создается с помощью конструктора. Логическая переменная **isRunning** будет следить за состоянием потока. В начале просмотра страницы переменной **isRunning** присваивается значение **true**, и с помощью метода **start()** запускается поток runner. Метод **start()** сообщает потоку о том, что следует запустить метод **run()**.

В методе **run()** выполняются действия: приостанавливается текущий поток на 300 миллисекунд, выбирается следующее изображение, перерисовывается изображение в области апплета. Чередованием этих действий достигается эффект движения как баскетболиста, так и мяча.

Для приостановки текущего процесса на указанный промежуток времени используется метод **sleep()**, который может завершиться исключительной ситуацией **InterruptedException**, если «заснувший» поток будет «разбужен».



Для приостановки текущего процесса на указанный промежуток времени используется метод sleep...

Анимация будет приостановлена, если произошло скрытие станицы с апплетом, в этом случае логическая переменная **IsRunning()** получит значение **false**. В листинге 7 представлен текст программы, использующей поток для получения анимационного эффекта.

На рис. 6 изображен один из кадров, обеспечивающих анимационный эффект. Через 300 миллисекунд загрузится другое изображение, тем самым будет достигнут эффект движения.

ДВИЖЕНИЕ СЛЕВА НАПРАВО И ОБРАТНО

Усложним предыдущую задачу. Рассмотрим случай, когда изображение (картинка) осуществляет движение по заданной траектории, пусть для простоты сначала происходит движение слева направо. При достижении правого края происходит «поворот» в обратную сторону, и далее осуществляется движение справа налево. При достижении левого края осуществляется поворот в обратную сторону и т. д.

При выводе изображения надо фиксировать текущие координаты. Так как

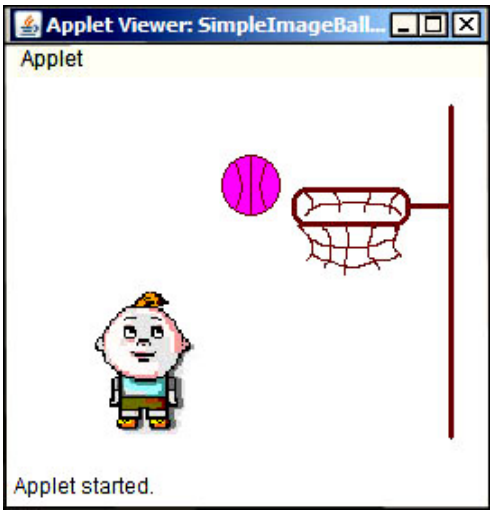


Рис. 6. Анимационный эффект (слайд-шоу)

Листинг 7. Слайд-шоу

```
import java.applet.Applet;
import java.awt.*;
public class SimpleImageBall1 extends Applet implements Runnable {
// количество изображений
     int pictLength = 10;
    Image[] pict = new Image[pictLength];
// номер текущего изображения
    int currentNum = 0;
// новый поток
    Thread runner = new Thread(this);
// используется  для остановки анимации
    boolean isRunning;
// инициализация
    public void init() {
      resize(250,250);
// формирование массива изображений
      for (int i = 0; i < pictLength; i++) {
        pict[i] = getImage(getDocumentBase(),
                                  "../img/bascet/ball" + i + ".gif");
      }
    }
// начало просмотра  страницы с апплетом
    public void start() {
        // запуск анимации
        isRunning = true;
        // при вызове этого метода запускается run()
        runner.start();
    }
// скрытие страницы с апплетом, например, при переключении
// на другое окно и т.п.
    public void stop() {
        // остановка анимации
        isRunning = false;
    }
 // запуск потока
    public void run() {
        try {
            while (isRunning) {
                Thread.sleep(300);
                     // меняется  номер текущего изображения
                     // и перерисовывается апплет
                currentNum++;
                if (currentNum >9) currentNum=0;
                repaint();
            }
       } catch(InterruptedException e) {}
   }
    public void update(Graphics g) {
        paint(g);
    }
//рисование
    public void paint(Graphics g) {
        g.drawImage(pict[currentNum], 0, 0, this);
    }
}
```

Листинг 8. Движение по прямой справа налево и обратно

```
import java.applet.Applet;
import java.awt.*;
public class SimpleImageSequence2Applet extends Applet
      implements Runnable {
// число изображений в массиве
    int pictLength = 20;
    Image[] pict = new Image[pictLength];
    Image background; // фоновое изображение
    int currentNum = 0; // номер текущего изображения
    int x = 0; // координата, с которой выводится изображение
// новый поток
    Thread runner = new Thread(this);
    boolean isRunning;
// действия при инициализации
    public void init() {
        // изображение в качестве фона
        background = getImage(getDocumentBase(), "../img/grass.jpg");
        // Для анимации используются изображения с прозрачным фоном
        for (int i = 0; i < pictLength; i++) {
           pict[i] = getImage(getDocumentBase(),
                                   "../img/sequence/dog" + i + ".gif");
        }
        resize(540,110);// размеры окна апплета
    }
// открытие страницы
    public void start() {
        isRunning = true;
        runner.start();
    }
// скрытие страницы
    public void stop() {
        isRunning = false;
    }
// поток
    public void run() {
        try {
            while (isRunning) {
                Thread.sleep(300);
                currentNum = (currentNum + 1) % pictLength;
                 // изменение номера изображения и текущей координаты x
                 // в крайних положениях поворот в обратнои направлении
                if (currentNum > 0 && currentNum < 9) {
                    // движение вправо
                    x = x+50;
                } else if (currentNum > 10 && currentNum < 19) {
                    // движение влево
                    x = x-50;
                }
                repaint();
            }
        } catch(InterruptedException e) {}
    }
// прорисовка
    public void paint(Graphics g) {
        g.drawImage(background, 0, 0, this);
        g.drawImage(pict[currentNum], x, 0, 140, 110,  this);
    }
}
```

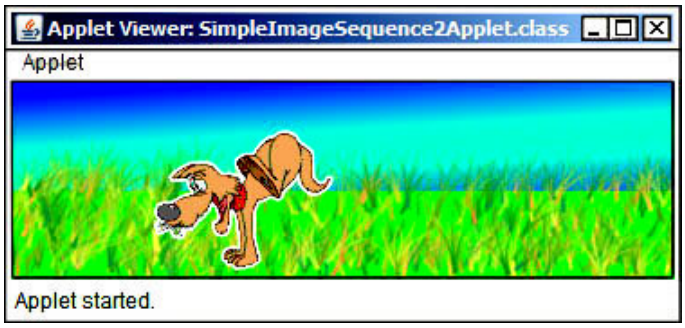
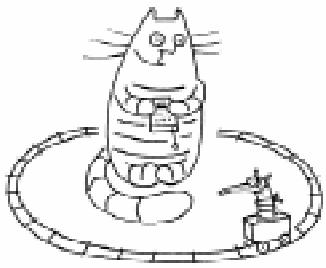


Рис. 7. Движение слева направо и обратно



... изображение ... осуществляет движение по заданной траектории.

происходит движение слева направо и обратно, то изменяться будет только одна координата.

В программе одно изображение будет использоваться в качестве фонового. Изображения, на основе которых реализуется анимационный эффект, формируются в массив изображений. Десять изображений могут быть использованы для движения слева направо, десять изображений – для движения справа налево.

После выбора изображения, которое требуется вывести, вычисляются координаты области, с которых осуществляется вывод. В данном случае меняется только координата *x*: увеличивается при движении направо, уменьшается при движении налево. По индексу выводимого изображения определяется направление движения и соответственно, новые координаты для вывода изображения. Анимационный эффект реализуется с помощью потока. В листинге 8 представлен текст исходной программы.

На рис. 7 изображен один из кадров, обеспечивающих анимационный эффект.

УПРАЖНЕНИЯ

1. Создайте программу, иллюстрирующую работу светофора.
2. Создайте программу, иллюстрирующую движение солнца по небосклону.
3. Создайте программу, иллюстрирующую песочные часы.
4. В следующих заданиях требуется создать иллюзию движения. Анимация основана на смене кадров, каждый из которых соответствует очередному положению движущегося объекта. Требуется написать программу, иллюстрирующую пословицу.

- Своя ноша рук не тянет.
- Век живи – век учись.
- Слышал звон, да не знает, где он.
- Старый конь борозды не испортит.
- Новая метла чисто метет.
- Дурная голова ногам покоя не дает.
- Где тонко, там и рвется.
- Кто много веселится, тому некогда учиться.
- Нужда и голод выгоняют на холод.
- Чему быть, того не миновать.
- С волками жить, по-волчьи выть.
- Пеший конному не товарищ.

Дмитриева Марина Валерьевна, доцент кафедры информатики математико-механического факультета Санкт-Петербургского государственного университета.