**Moving sprites**

In this part of the Java 2D games tutorial we will work with sprites.

 The term *sprite* has several meanings. It is used to denote an image or an animation in a scene.

It is also used to represent any movable object in a game. Also one of the meanings is the code that encapsulates a character in a game. In our tutorial by using sprite we refer to a movable object or its Java class.

**Moving sprite**

In the first example we have a spacecraft. We can move the spacecraft on the board using the cursor keys.

Craft.java

package com.zetcode;

import java.awt.Image;

import java.awt.event.KeyEvent;

import javax.swing.ImageIcon;

public class Craft {

private int dx;

private int dy;

private int x;

private int y;

private Image image;

public Craft() {

initCraft();

}

private void initCraft() {

ImageIcon ii = new ImageIcon("craft.png");

image = ii.getImage();

x = 40;

y = 60;

}

public void move() {

x += dx;

y += dy;

}

public int getX() {

return x;

}

public int getY() {

return y;

}

public Image getImage() {

return image;

}

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (key == KeyEvent.VK\_LEFT) {

dx = -1;

}

if (key == KeyEvent.VK\_RIGHT) {

dx = 1;

}

if (key == KeyEvent.VK\_UP) {

dy = -1;

}

if (key == KeyEvent.VK\_DOWN) {

dy = 1;

}

}

public void keyReleased(KeyEvent e) {

int key = e.getKeyCode();

if (key == KeyEvent.VK\_LEFT) {

dx = 0;

}

if (key == KeyEvent.VK\_RIGHT) {

dx = 0;

}

if (key == KeyEvent.VK\_UP) {

dy = 0;

}

if (key == KeyEvent.VK\_DOWN) {

dy = 0;

}

}

}

This class represents a spacecraft. In this class we keep the image of the sprite and the coordinates of the sprite. The keyPressed() and keyReleased() methods control whether the sprite is moving.

public void move() {

x += dx;

y += dy;

}

The move() method changes the coordinates of the sprite. These x and y values are used in thepaintComponent() method to draw the image of the sprite.

if (key == KeyEvent.VK\_LEFT) {

dx = 0;

}

When we release the left cursor key, we set the dx variable to zero. The spacecraft will stop moving.

Board.java

package com.zetcode;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import javax.swing.JPanel;

import javax.swing.Timer;

public class Board extends JPanel implements ActionListener {

private Timer timer;

private Craft craft;

private final int DELAY = 10;

public Board() {

initBoard();

}

private void initBoard() {

addKeyListener(new TAdapter());

setFocusable(true);

setBackground(Color.BLACK);

craft = new Craft();

timer = new Timer(DELAY, this);

timer.start();

}

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

doDrawing(g);

Toolkit.getDefaultToolkit().sync();

}

private void doDrawing(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(craft.getImage(), craft.getX(), craft.getY(), this);

}

@Override

public void actionPerformed(ActionEvent e) {

craft.move();

repaint();

}

private class TAdapter extends KeyAdapter {

@Override

public void keyReleased(KeyEvent e) {

craft.keyReleased(e);

}

@Override

public void keyPressed(KeyEvent e) {

craft.keyPressed(e);

}

}

}

This is the Board class.

private void doDrawing(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(craft.getImage(), craft.getX(), craft.getY(), this);

}

In the doDrawing() method, we draw the spacecraft with the drawImage() method. We get the image and the coordinates from the sprite class.

@Override

public void actionPerformed(ActionEvent e) {

craft.move();

repaint();

}

The actionPerformed() method is called every DELAY ms. We move the sprite and repaint the board.

private class TAdapter extends KeyAdapter {

@Override

public void keyReleased(KeyEvent e) {

craft.keyReleased(e);

}

@Override

public void keyPressed(KeyEvent e) {

craft.keyPressed(e);

}

}

In the Board class we listen for key events. The overridden methods of the KeyAdapter class delegate the processing to the methods of the Craft class.

MovingSpriteEx.java

package com.zetcode;

import java.awt.EventQueue;

import javax.swing.JFrame;

public class MovingSpriteEx extends JFrame {

public MovingSpriteEx() {

initUI();

}

private void initUI() {

add(new Board());

setSize(400, 300);

setResizable(false);

setTitle("Moving sprite");

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

MovingSpriteEx ex = new MovingSpriteEx();

ex.setVisible(true);

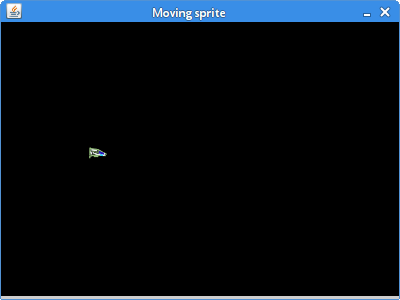
}

});

}

}

This is the main class.

Figure: Moving sprite

**Shooting missiles**

In the next example we will add another sprite type to our example—a missile. The missiles are launched with the Space key.

Sprite.java

package com.zetcode;

import java.awt.Image;

import javax.swing.ImageIcon;

public class Sprite {

protected int x;

protected int y;

protected int width;

protected int height;

protected boolean vis;

protected Image image;

public Sprite(int x, int y) {

this.x = x;

this.y = y;

vis = true;

}

protected void loadImage(String imageName) {

ImageIcon ii = new ImageIcon(imageName);

image = ii.getImage();

}

protected void getImageDimensions() {

width = image.getWidth(null);

height = image.getHeight(null);

}

public Image getImage() {

return image;

}

public int getX() {

return x;

}

public int getY() {

return y;

}

public boolean isVisible() {

return vis;

}

public void setVisible(Boolean visible) {

vis = visible;

}

}

The Sprite class shares common code from the Missile and Craft classes.

public Sprite(int x, int y) {

this.x = x;

this.y = y;

vis = true;

}

The constructor initiates the x and y coordiates and the vis variable.

Missile.java

package com.zetcode;

public class Missile extends Sprite {

private final int BOARD\_WIDTH = 390;

private final int MISSILE\_SPEED = 2;

public Missile(int x, int y) {

super(x, y);

initMissile();

}

private void initMissile() {

loadImage("missile.png");

getImageDimensions();

}

public void move() {

x += MISSILE\_SPEED;

if (x > BOARD\_WIDTH) {

vis = false;

}

}

}

Here we have a new sprite called Missile.

public void move() {

x += MISSILE\_SPEED;

if (x > BOARD\_WIDTH) {

vis = false;

}

}

The missile moves at constant speed. When it hits the right border of the Board, it becomes invisible. It is then removed from the list of missiles.

Craft.java

package com.zetcode;

import java.awt.event.KeyEvent;

import java.util.ArrayList;

public class Craft extends Sprite {

private int dx;

private int dy;

private ArrayList missiles;

public Craft(int x, int y) {

super(x, y);

initCraft();

}

private void initCraft() {

missiles = new ArrayList();

loadImage("craft.png");

getImageDimensions();

}

public void move() {

x += dx;

y += dy;

}

public ArrayList getMissiles() {

return missiles;

}

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (key == KeyEvent.VK\_SPACE) {

fire();

}

if (key == KeyEvent.VK\_LEFT) {

dx = -1;

}

if (key == KeyEvent.VK\_RIGHT) {

dx = 1;

}

if (key == KeyEvent.VK\_UP) {

dy = -1;

}

if (key == KeyEvent.VK\_DOWN) {

dy = 1;

}

}

public void fire() {

missiles.add(new Missile(x + width, y + height / 2));

}

public void keyReleased(KeyEvent e) {

int key = e.getKeyCode();

if (key == KeyEvent.VK\_LEFT) {

dx = 0;

}

if (key == KeyEvent.VK\_RIGHT) {

dx = 0;

}

if (key == KeyEvent.VK\_UP) {

dy = 0;

}

if (key == KeyEvent.VK\_DOWN) {

dy = 0;

}

}

}

This is the Craft class.

if (key == KeyEvent.VK\_SPACE) {

fire();

}

If we press the Space key, we fire.

public void fire() {

missiles.add(new Missile(x + width, y + height / 2));

}

The fire() method creates a new Missile object and adds it to the missiles ArrayList.

public ArrayList getMissiles() {

return missiles;

}

The getMissiles() method returns the ArrayList of missiles. It is called from the Board class.

Board.java

package com.zetcode;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.util.ArrayList;

import javax.swing.JPanel;

import javax.swing.Timer;

public class Board extends JPanel implements ActionListener {

private final int ICRAFT\_X = 40;

private final int ICRAFT\_Y = 60;

private final int DELAY = 10;

private Timer timer;

private Craft craft;

public Board() {

initBoard();

}

private void initBoard() {

addKeyListener(new TAdapter());

setFocusable(true);

setBackground(Color.BLACK);

setDoubleBuffered(true);

craft = new Craft(ICRAFT\_X, ICRAFT\_Y);

timer = new Timer(DELAY, this);

timer.start();

}

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

doDrawing(g);

Toolkit.getDefaultToolkit().sync();

}

private void doDrawing(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(craft.getImage(), craft.getX(),

craft.getY(), this);

ArrayList ms = craft.getMissiles();

for (Object m1 : ms) {

Missile m = (Missile) m1;

g2d.drawImage(m.getImage(), m.getX(),

m.getY(), this);

}

}

@Override

public void actionPerformed(ActionEvent e) {

updateMissiles();

updateCraft();

repaint();

}

private void updateMissiles() {

ArrayList ms = craft.getMissiles();

for (int i = 0; i < ms.size(); i++) {

Missile m = (Missile) ms.get(i);

if (m.isVisible()) {

m.move();

} else {

ms.remove(i);

}

}

}

private void updateCraft() {

craft.move();

}

private class TAdapter extends KeyAdapter {

@Override

public void keyReleased(KeyEvent e) {

craft.keyReleased(e);

}

@Override

public void keyPressed(KeyEvent e) {

craft.keyPressed(e);

}

}

}

This is the Board class.

private void doDrawing(Graphics g) {

Graphics2D g2d = (Graphics2D) g;

g2d.drawImage(craft.getImage(), craft.getX(),

craft.getY(), this);

ArrayList ms = craft.getMissiles();

for (Object m1 : ms) {

Missile m = (Missile) m1;

g2d.drawImage(m.getImage(), m.getX(),

m.getY(), this);

}

}

In the doDrawing() method, we draw the craft and all the available missiles.

private void updateMissiles() {

ArrayList ms = craft.getMissiles();

for (int i = 0; i < ms.size(); i++) {

Missile m = (Missile) ms.get(i);

if (m.isVisible()) {

m.move();

} else {

ms.remove(i);

}

}

}

In the updateMissiles() method we parse all missiles from the missiles list. Depending on what theisVisible() method returns, we either move the missile or remove it from the container.

ShootingMissilesEx.java

package com.zetcode;

import java.awt.EventQueue;

import javax.swing.JFrame;

public class ShootingMissilesEx extends JFrame {

public ShootingMissilesEx() {

initUI();

}

private void initUI() {

add(new Board());

setSize(400, 300);

setResizable(false);

setTitle("Shooting missiles");

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

ShootingMissilesEx ex = new ShootingMissilesEx();

ex.setVisible(true);

}

});

}

}

Finally, this is the main class.

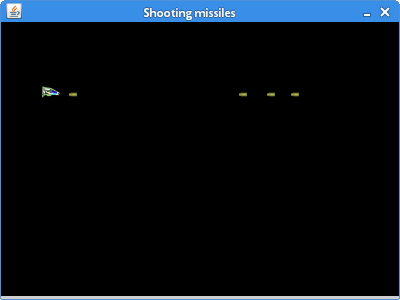


Figure: Shooting missiles