**Collision detection**

In this part of the Java 2D games tutorial we will talk about collision detection.

Many games need to handle collisions, especially arcade games. Simply said, we need to detect when two objects collide on screen.

In the next code example, we will expand the previous example. We add a new Alien sprite. We will detect two types of collisions: when the missile hits an alien ship and when our spacecraft collides with an alien.

**Shooting aliens**

In the example, we have a spacecraft and aliens. We can move the spacecraft on the board using the cursor keys. Missiles destroying aliens are launched with the spacebar key.

Sprite.java

package com.zetcode;

import java.awt.Image;

import java.awt.Rectangle;

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 getImageDimensions() {

width = image.getWidth(null);

height = image.getHeight(null);

}

protected void loadImage(String imageName) {

ImageIcon ii = new ImageIcon(imageName);

image = ii.getImage();

}

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;

}

public Rectangle getBounds() {

return new Rectangle(x, y, width, height);

}

}

The code that can be shared by all sprites (a craft, an alien, and a missile) is placed in the Spriteclass.

public Rectangle getBounds() {

return new Rectangle(x, y, width, height);

}

The getBounds() method returns the bounding rectangle of the sprite image. We need the bounds in collision detection.

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<Missile> 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;

if (x < 1) {

x = 1;

}

if (y < 1) {

y = 1;

}

}

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 class represents a spacecraft.

private ArrayList<Missile> missiles;

All the missiles fired by the spacecraft are stored in the missiles list.

public void fire() {

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

}

When we fire a missile, a new Missile object is added to the missiles list. It is retained in the list until it collides with an alien or goes out of the window.

Board.java

package com.zetcode;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.FontMetrics;

import java.awt.Graphics;

import java.awt.Rectangle;

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 Timer timer;

private Craft craft;

private ArrayList<Alien> aliens;

private boolean ingame;

private final int ICRAFT\_X = 40;

private final int ICRAFT\_Y = 60;

private final int B\_WIDTH = 400;

private final int B\_HEIGHT = 300;

private final int DELAY = 15;

private final int[][] pos = {

{2380, 29}, {2500, 59}, {1380, 89},

{780, 109}, {580, 139}, {680, 239},

{790, 259}, {760, 50}, {790, 150},

{980, 209}, {560, 45}, {510, 70},

{930, 159}, {590, 80}, {530, 60},

{940, 59}, {990, 30}, {920, 200},

{900, 259}, {660, 50}, {540, 90},

{810, 220}, {860, 20}, {740, 180},

{820, 128}, {490, 170}, {700, 30}

};

public Board() {

initBoard();

}

private void initBoard() {

addKeyListener(new TAdapter());

setFocusable(true);

setBackground(Color.BLACK);

ingame = true;

setPreferredSize(new Dimension(B\_WIDTH, B\_HEIGHT));

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

initAliens();

timer = new Timer(DELAY, this);

timer.start();

}

public void initAliens() {

aliens = new ArrayList<>();

for (int[] p : pos) {

aliens.add(new Alien(p[0], p[1]));

}

}

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

if (ingame) {

drawObjects(g);

} else {

drawGameOver(g);

}

Toolkit.getDefaultToolkit().sync();

}

private void drawObjects(Graphics g) {

if (craft.isVisible()) {

g.drawImage(craft.getImage(), craft.getX(), craft.getY(),

this);

}

ArrayList<Missile> ms = craft.getMissiles();

for (Missile m : ms) {

if (m.isVisible()) {

g.drawImage(m.getImage(), m.getX(), m.getY(), this);

}

}

for (Alien a : aliens) {

if (a.isVisible()) {

g.drawImage(a.getImage(), a.getX(), a.getY(), this);

}

}

g.setColor(Color.WHITE);

g.drawString("Aliens left: " + aliens.size(), 5, 15);

}

private void drawGameOver(Graphics g) {

String msg = "Game Over";

Font small = new Font("Helvetica", Font.BOLD, 14);

FontMetrics fm = getFontMetrics(small);

g.setColor(Color.white);

g.setFont(small);

g.drawString(msg, (B\_WIDTH - fm.stringWidth(msg)) / 2,

B\_HEIGHT / 2);

}

@Override

public void actionPerformed(ActionEvent e) {

inGame();

updateCraft();

updateMissiles();

updateAliens();

checkCollisions();

repaint();

}

private void inGame() {

if (!ingame) {

timer.stop();

}

}

private void updateCraft() {

if (craft.isVisible()) {

craft.move();

}

}

private void updateMissiles() {

ArrayList<Missile> ms = craft.getMissiles();

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

Missile m = ms.get(i);

if (m.isVisible()) {

m.move();

} else {

ms.remove(i);

}

}

}

private void updateAliens() {

if (aliens.isEmpty()) {

ingame = false;

return;

}

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

Alien a = aliens.get(i);

if (a.isVisible()) {

a.move();

} else {

aliens.remove(i);

}

}

}

public void checkCollisions() {

Rectangle r3 = craft.getBounds();

for (Alien alien : aliens) {

Rectangle r2 = alien.getBounds();

if (r3.intersects(r2)) {

craft.setVisible(false);

alien.setVisible(false);

ingame = false;

}

}

ArrayList<Missile> ms = craft.getMissiles();

for (Missile m : ms) {

Rectangle r1 = m.getBounds();

for (Alien alien : aliens) {

Rectangle r2 = alien.getBounds();

if (r1.intersects(r2)) {

m.setVisible(false);

alien.setVisible(false);

}

}

}

}

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 final int[][] pos = {

{2380, 29}, {2500, 59}, {1380, 89},

{780, 109}, {580, 139}, {680, 239},

{790, 259}, {760, 50}, {790, 150},

{980, 209}, {560, 45}, {510, 70},

{930, 159}, {590, 80}, {530, 60},

{940, 59}, {990, 30}, {920, 200},

{900, 259}, {660, 50}, {540, 90},

{810, 220}, {860, 20}, {740, 180},

{820, 128}, {490, 170}, {700, 30}

};

These are the initial positions of alien ships.

public void initAliens() {

aliens = new ArrayList<>();

for (int[] p : pos) {

aliens.add(new Alien(p[0], p[1]));

}

}

The initAliens() method creates a list of alien objects. The aliens take their initial positions from the pos array.

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

if (ingame) {

drawObjects(g);

} else {

drawGameOver(g);

}

Toolkit.getDefaultToolkit().sync();

}

Inside the paintComponent() method, we either draw game sprites or write the game over message. This depends on the ingame variable.

private void drawObjects(Graphics g) {

if (craft.isVisible()) {

g.drawImage(craft.getImage(), craft.getX(), craft.getY(),

this);

}

...

}

The drawObjects() method draws game sprites on the window. First, we draw the craft sprite.

for (Alien a : aliens) {

if (a.isVisible()) {

g.drawImage(a.getImage(), a.getX(), a.getY(), this);

}

}

In this loop we draw all aliens; they are drawn only if they have not been previously destroyed. This is checked by the isVisible() method.

g.setColor(Color.WHITE);

g.drawString("Aliens left: " + aliens.size(), 5, 15);

In the top-left corner of the window, we draw how many aliens are left.

private void drawGameOver(Graphics g) {

String msg = "Game Over";

Font small = new Font("Helvetica", Font.BOLD, 14);

FontMetrics fm = getFontMetrics(small);

g.setColor(Color.white);

g.setFont(small);

g.drawString(msg, (B\_WIDTH - fm.stringWidth(msg)) / 2,

B\_HEIGHT / 2);

}

The drawGameOver() draws a game over message in the middle of the window. The message is displayed at the end of the game, either when we destroy all alien ships or when we collide with one of them.

@Override

public void actionPerformed(ActionEvent e) {

inGame();

updateCraft();

updateMissiles();

updateAliens();

checkCollisions();

repaint();

}

Each action event represents one game cycle. The game logic is factored into specific methods. For instance, the updateMissiles() moves all the available missiles.

private void updateAliens() {

if (aliens.isEmpty()) {

ingame = false;

return;

}

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

Alien a = aliens.get(i);

if (a.isVisible()) {

a.move();

} else {

aliens.remove(i);

}

}

}

Inside the updateAliens() method, we first check if there are any alien objects left in the aliens list. The game is finished if the list is empty. If it is not empty, we go trough the list and move all its items. The destroyed aliens are removed from the list.

public void checkCollisions() {

Rectangle r3 = craft.getBounds();

for (Alien alien : aliens) {

Rectangle r2 = alien.getBounds();

if (r3.intersects(r2)) {

craft.setVisible(false);

alien.setVisible(false);

ingame = false;

}

}

...

}

The checkCollisions() method checks for possible collisions. First, we check if the craft object collides with any of the alien objects. We get the rectangles of the objects with the getBounds()method. The intersects() method checks if the two rectangles intersect.

ArrayList<Missile> ms = craft.getMissiles();

for (Missile m : ms) {

Rectangle r1 = m.getBounds();

for (Alien alien : aliens) {

Rectangle r2 = alien.getBounds();

if (r1.intersects(r2)) {

m.setVisible(false);

alien.setVisible(false);

}

}

}

This code checks the collisions between missiles and aliens.

Alien.java

package com.zetcode;

public class Alien extends Sprite {

private final int INITIAL\_X = 400;

public Alien(int x, int y) {

super(x, y);

initAlien();

}

private void initAlien() {

loadImage("alien.png");

getImageDimensions();

}

public void move() {

if (x < 0) {

x = INITIAL\_X;

}

x -= 1;

}

}

This is the Alien class.

public void move() {

if (x < 0) {

x = INITIAL\_X;

}

x -= 1;

}

Aliens return to the screen on the right side after they have disappeared on the left.

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;

}

}

This is the Missile class.

public void move() {

x += MISSILE\_SPEED;

if (x > BOARD\_WIDTH)

vis = false;

}

Missiles move in one direction only. They disappear after they reach the right window border.

CollisionEx.java

package com.zetcode;

import java.awt.EventQueue;

import javax.swing.JFrame;

public class CollisionEx extends JFrame {

public CollisionEx() {

initUI();

}

private void initUI() {

add(new Board());

setResizable(false);

pack();

setTitle("Collision");

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

@Override

public void run() {

CollisionEx ex = new CollisionEx();

ex.setVisible(true);

}

});

}

}

Finally, this is the main class.

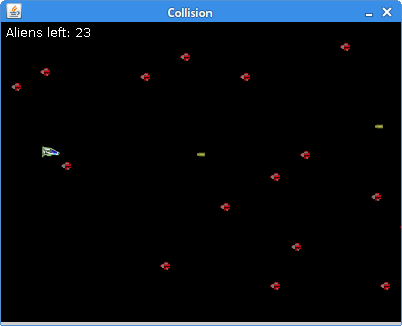


Figure: Shooting aliens