**Space Invaders**

In this part of the Java 2D games tutorial we will create a simple Space Invaders game clone.

*Space Invaders* is an arcade video game designed by *Tomohiro Nishikado*. It was first released in 1978. The player controls a cannon. He is about to save the Earth from invasion of evil space invaders.

**Development**

In our Java clone we have 24 invaders. These aliens heavily shell the ground. When the player shoots a missile, he can shoot another one only when it hits an alien or the top of the Board. The player shoots with the Alt key. Aliens launch randomly their bombs. Each alien shoots a bomb only after the previous one hits the bottom.

SpaceInvaders.java

package spaceinvaders;

import javax.swing.JFrame;

public class SpaceInvaders extends JFrame implements Commons {

public SpaceInvaders()

{

add(new Board());

setTitle("Space Invaders");

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setSize(BOARD\_WIDTH, BOARD\_HEIGTH);

setLocationRelativeTo(null);

setVisible(true);

setResizable(false);

}

public static void main(String[] args) {

new SpaceInvaders();

}

}

This is the main class.

Commons.java

package spaceinvaders;

public interface Commons {

public static final int BOARD\_WIDTH = 358;

public static final int BOARD\_HEIGTH = 350;

public static final int GROUND = 290;

public static final int BOMB\_HEIGHT = 5;

public static final int ALIEN\_HEIGHT = 12;

public static final int ALIEN\_WIDTH = 12;

public static final int BORDER\_RIGHT = 30;

public static final int BORDER\_LEFT = 5;

public static final int GO\_DOWN = 15;

public static final int NUMBER\_OF\_ALIENS\_TO\_DESTROY = 24;

public static final int CHANCE = 5;

public static final int DELAY = 17;

public static final int PLAYER\_WIDTH = 15;

public static final int PLAYER\_HEIGHT = 10;

}

The Commons.java file has some common constants. They are self-explanatory.

Alien.java

package spaceinvaders;

import javax.swing.ImageIcon;

public class Alien extends Sprite {

private Bomb bomb;

private final String shot = "../spacepix/alien.png";

public Alien(int x, int y) {

this.x = x;

this.y = y;

bomb = new Bomb(x, y);

ImageIcon ii = new ImageIcon(this.getClass().getResource(shot));

setImage(ii.getImage());

}

public void act(int direction) {

this.x += direction;

}

public Bomb getBomb() {

return bomb;

}

public class Bomb extends Sprite {

private final String bomb = "../spacepix/bomb.png";

private boolean destroyed;

public Bomb(int x, int y) {

setDestroyed(true);

this.x = x;

this.y = y;

ImageIcon ii = new ImageIcon(this.getClass().getResource(bomb));

setImage(ii.getImage());

}

public void setDestroyed(boolean destroyed) {

this.destroyed = destroyed;

}

public boolean isDestroyed() {

return destroyed;

}

}

}

This is the Alien sprite. Each alien has an inner Bomb class.

public void act(int direction) {

this.x += direction;

}

The act() method is called from the Board class. It is used to position an alien in horizontal direction.

public Bomb getBomb() {

return bomb;

}

The getBomb() method is called, when the alien is about to drop a bomb.

Player.java

package spaceinvaders;

import java.awt.event.KeyEvent;

import javax.swing.ImageIcon;

public class Player extends Sprite implements Commons{

private final int START\_Y = 280;

private final int START\_X = 270;

private final String player = "../spacepix/player.png";

private int width;

public Player() {

ImageIcon ii = new ImageIcon(this.getClass().getResource(player));

width = ii.getImage().getWidth(null);

setImage(ii.getImage());

setX(START\_X);

setY(START\_Y);

}

public void act() {

x += dx;

if (x <= 2)

x = 2;

if (x >= BOARD\_WIDTH - 2\*width)

x = BOARD\_WIDTH - 2\*width;

}

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (key == KeyEvent.VK\_LEFT)

{

dx = -2;

}

if (key == KeyEvent.VK\_RIGHT)

{

dx = 2;

}

}

public void keyReleased(KeyEvent e) {

int key = e.getKeyCode();

if (key == KeyEvent.VK\_LEFT)

{

dx = 0;

}

if (key == KeyEvent.VK\_RIGHT)

{

dx = 0;

}

}

}

This is the Player sprite. We control the cannon with the cursor keys.

private final int START\_Y = 280;

private final int START\_X = 270;

These are the initial coordinates of the player sprite.

if (key == KeyEvent.VK\_LEFT)

{

dx = -2;

}

If we press the left cursor key, the dx variable is set to -2. Next time the act() method is called, the player moves to the left.

if (key == KeyEvent.VK\_LEFT)

{

dx = 0;

}

if (key == KeyEvent.VK\_RIGHT)

{

dx = 0;

}

If we release the left or the right cursor, the dx variable is set to zero. The player sprite stops moving.

Shot.java

package spaceinvaders;

import javax.swing.ImageIcon;

public class Shot extends Sprite {

private String shot = "../spacepix/shot.png";

private final int H\_SPACE = 6;

private final int V\_SPACE = 1;

public Shot() {

}

public Shot(int x, int y) {

ImageIcon ii = new ImageIcon(this.getClass().getResource(shot));

setImage(ii.getImage());

setX(x + H\_SPACE);

setY(y - V\_SPACE);

}

}

This is the Shot sprite. The shot is triggered with the ALT key. The H\_SPACE and the V\_SPACE constants are used to position the missile appropriately.

Sprite.java

package spaceinvaders;

import java.awt.Image;

public class Sprite {

private boolean visible;

private Image image;

protected int x;

protected int y;

protected boolean dying;

protected int dx;

public Sprite() {

visible = true;

}

public void die() {

visible = false;

}

public boolean isVisible() {

return visible;

}

protected void setVisible(boolean visible) {

this.visible = visible;

}

public void setImage(Image image) {

this.image = image;

}

public Image getImage() {

return image;

}

public void setX(int x) {

this.x = x;

}

public void setY(int y) {

this.y = y;

}

public int getY() {

return y;

}

public int getX() {

return x;

}

public void setDying(boolean dying) {

this.dying = dying;

}

public boolean isDying() {

return this.dying;

}

}

This is the basic Sprite class. Other sprites inherit from it. It has some common functionality.

Board.java

package spaceinvaders;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.FontMetrics;

import java.awt.Graphics;

import java.awt.Toolkit;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.Random;

import javax.swing.ImageIcon;

import javax.swing.JPanel;

public class Board extends JPanel implements Runnable, Commons {

private Dimension d;

private ArrayList aliens;

private Player player;

private Shot shot;

private int alienX = 150;

private int alienY = 5;

private int direction = -1;

private int deaths = 0;

private boolean ingame = true;

private final String expl = "../spacepix/explosion.png";

private final String alienpix = "../spacepix/alien.png";

private String message = "Game Over";

private Thread animator;

public Board()

{

addKeyListener(new TAdapter());

setFocusable(true);

d = new Dimension(BOARD\_WIDTH, BOARD\_HEIGTH);

setBackground(Color.black);

gameInit();

setDoubleBuffered(true);

}

public void addNotify() {

super.addNotify();

gameInit();

}

public void gameInit() {

aliens = new ArrayList();

ImageIcon ii = new ImageIcon(this.getClass().getResource(alienpix));

for (int i=0; i < 4; i++) {

for (int j=0; j < 6; j++) {

Alien alien = new Alien(alienX + 18\*j, alienY + 18\*i);

alien.setImage(ii.getImage());

aliens.add(alien);

}

}

player = new Player();

shot = new Shot();

if (animator == null || !ingame) {

animator = new Thread(this);

animator.start();

}

}

public void drawAliens(Graphics g)

{

Iterator it = aliens.iterator();

while (it.hasNext()) {

Alien alien = (Alien) it.next();

if (alien.isVisible()) {

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

}

if (alien.isDying()) {

alien.die();

}

}

}

public void drawPlayer(Graphics g) {

if (player.isVisible()) {

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

}

if (player.isDying()) {

player.die();

ingame = false;

}

}

public void drawShot(Graphics g) {

if (shot.isVisible())

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

}

public void drawBombing(Graphics g) {

Iterator i3 = aliens.iterator();

while (i3.hasNext()) {

Alien a = (Alien) i3.next();

Alien.Bomb b = a.getBomb();

if (!b.isDestroyed()) {

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

}

}

}

public void paint(Graphics g)

{

super.paint(g);

g.setColor(Color.black);

g.fillRect(0, 0, d.width, d.height);

g.setColor(Color.green);

if (ingame) {

g.drawLine(0, GROUND, BOARD\_WIDTH, GROUND);

drawAliens(g);

drawPlayer(g);

drawShot(g);

drawBombing(g);

}

Toolkit.getDefaultToolkit().sync();

g.dispose();

}

public void gameOver()

{

Graphics g = this.getGraphics();

g.setColor(Color.black);

g.fillRect(0, 0, BOARD\_WIDTH, BOARD\_HEIGTH);

g.setColor(new Color(0, 32, 48));

g.fillRect(50, BOARD\_WIDTH/2 - 30, BOARD\_WIDTH-100, 50);

g.setColor(Color.white);

g.drawRect(50, BOARD\_WIDTH/2 - 30, BOARD\_WIDTH-100, 50);

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

FontMetrics metr = this.getFontMetrics(small);

g.setColor(Color.white);

g.setFont(small);

g.drawString(message, (BOARD\_WIDTH - metr.stringWidth(message))/2,

BOARD\_WIDTH/2);

}

public void animationCycle() {

if (deaths == NUMBER\_OF\_ALIENS\_TO\_DESTROY) {

ingame = false;

message = "Game won!";

}

// player

player.act();

// shot

if (shot.isVisible()) {

Iterator it = aliens.iterator();

int shotX = shot.getX();

int shotY = shot.getY();

while (it.hasNext()) {

Alien alien = (Alien) it.next();

int alienX = alien.getX();

int alienY = alien.getY();

if (alien.isVisible() && shot.isVisible()) {

if (shotX >= (alienX) &&

shotX <= (alienX + ALIEN\_WIDTH) &&

shotY >= (alienY) &&

shotY <= (alienY+ALIEN\_HEIGHT) ) {

ImageIcon ii =

new ImageIcon(getClass().getResource(expl));

alien.setImage(ii.getImage());

alien.setDying(true);

deaths++;

shot.die();

}

}

}

int y = shot.getY();

y -= 4;

if (y < 0)

shot.die();

else shot.setY(y);

}

// aliens

Iterator it1 = aliens.iterator();

while (it1.hasNext()) {

Alien a1 = (Alien) it1.next();

int x = a1.getX();

if (x >= BOARD\_WIDTH - BORDER\_RIGHT && direction != -1) {

direction = -1;

Iterator i1 = aliens.iterator();

while (i1.hasNext()) {

Alien a2 = (Alien) i1.next();

a2.setY(a2.getY() + GO\_DOWN);

}

}

if (x <= BORDER\_LEFT && direction != 1) {

direction = 1;

Iterator i2 = aliens.iterator();

while (i2.hasNext()) {

Alien a = (Alien)i2.next();

a.setY(a.getY() + GO\_DOWN);

}

}

}

Iterator it = aliens.iterator();

while (it.hasNext()) {

Alien alien = (Alien) it.next();

if (alien.isVisible()) {

int y = alien.getY();

if (y > GROUND - ALIEN\_HEIGHT) {

ingame = false;

message = "Invasion!";

}

alien.act(direction);

}

}

// bombs

Iterator i3 = aliens.iterator();

Random generator = new Random();

while (i3.hasNext()) {

int shot = generator.nextInt(15);

Alien a = (Alien) i3.next();

Alien.Bomb b = a.getBomb();

if (shot == CHANCE && a.isVisible() && b.isDestroyed()) {

b.setDestroyed(false);

b.setX(a.getX());

b.setY(a.getY());

}

int bombX = b.getX();

int bombY = b.getY();

int playerX = player.getX();

int playerY = player.getY();

if (player.isVisible() && !b.isDestroyed()) {

if ( bombX >= (playerX) &&

bombX <= (playerX+PLAYER\_WIDTH) &&

bombY >= (playerY) &&

bombY <= (playerY+PLAYER\_HEIGHT) ) {

ImageIcon ii =

new ImageIcon(this.getClass().getResource(expl));

player.setImage(ii.getImage());

player.setDying(true);

b.setDestroyed(true);;

}

}

if (!b.isDestroyed()) {

b.setY(b.getY() + 1);

if (b.getY() >= GROUND - BOMB\_HEIGHT) {

b.setDestroyed(true);

}

}

}

}

public void run() {

long beforeTime, timeDiff, sleep;

beforeTime = System.currentTimeMillis();

while (ingame) {

repaint();

animationCycle();

timeDiff = System.currentTimeMillis() - beforeTime;

sleep = DELAY - timeDiff;

if (sleep < 0)

sleep = 2;

try {

Thread.sleep(sleep);

} catch (InterruptedException e) {

System.out.println("interrupted");

}

beforeTime = System.currentTimeMillis();

}

gameOver();

}

private class TAdapter extends KeyAdapter {

public void keyReleased(KeyEvent e) {

player.keyReleased(e);

}

public void keyPressed(KeyEvent e) {

player.keyPressed(e);

int x = player.getX();

int y = player.getY();

if (ingame)

{

if (e.isAltDown()) {

if (!shot.isVisible())

shot = new Shot(x, y);

}

}

}

}

}

The main logic of the game is located in the Board class.

for (int i=0; i < 4; i++) {

for (int j=0; j < 6; j++) {

Alien alien = new Alien(alienX + 18\*j, alienY + 18\*i);

alien.setImage(ii.getImage());

aliens.add(alien);

}

}

player = new Player();

shot = new Shot();

In the gameInit() method we set up 24 aliens. The alien image size is 12x12px. We put 6px space among the aliens. We also create the player and the shot objects.

public void drawBombing(Graphics g) {

Iterator i3 = aliens.iterator();

while (i3.hasNext()) {

Alien a = (Alien) i3.next();

Alien.Bomb b = a.getBomb();

if (!b.isDestroyed()) {

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

}

}

}

The drawBombing() method draws bombs launched by the aliens.

if (ingame) {

g.drawLine(0, GROUND, BOARD\_WIDTH, GROUND);

drawAliens(g);

drawPlayer(g);

drawShot(g);

drawBombing(g);

}

Inside the paint() method, we draw the ground, the aliens, the player, the shot, and the bombs.

Next we will examine the animationCycle() method.

if (deaths == NUMBER\_OF\_ALIENS\_TO\_DESTROY) {

ingame = false;

message = "Game won!";

}

If we destroy all aliens, we win the game. (24 in this game)

if (alien.isVisible() && shot.isVisible()) {

if (shotX >= (alienX) &&

shotX <= (alienX + ALIEN\_WIDTH) &&

shotY >= (alienY) &&

shotY <= (alienY+ALIEN\_HEIGHT) ) {

ImageIcon ii =

new ImageIcon(getClass().getResource(expl));

alien.setImage(ii.getImage());

alien.setDying(true);

deaths++;

shot.die();

}

}

If the shot triggered by the player collides with an alien, the alien ship is destroyed. More precisely, the dying flag is set. We use it to display an explosion. The deaths variable increases and the shot sprite is destroyed.

if (x >= BOARD\_WIDTH - BORDER\_RIGHT && direction != -1) {

direction = -1;

Iterator i1 = aliens.iterator();

while (i1.hasNext()) {

Alien a2 = (Alien) i1.next();

a2.setY(a2.getY() + GO\_DOWN);

}

}

If the aliens reach the right end of the Board, they move down and change their direction to the left.

Iterator it = aliens.iterator();

while (it.hasNext()) {

Alien alien = (Alien) it.next();

if (alien.isVisible()) {

int y = alien.getY();

if (y > GROUND - ALIEN\_HEIGHT) {

ingame = false;

message = "Invasion!";

}

alien.act(direction);

}

}

This code moves aliens. If they reach the bottom, the invasion begins.

int shot = generator.nextInt(15);

Alien a = (Alien) i3.next();

Alien.Bomb b = a.getBomb();

if (shot == CHANCE && a.isVisible() && b.isDestroyed()) {

b.setDestroyed(false);

b.setX(a.getX());

b.setY(a.getY());

}

This is the code that determines whether the alien will drop a bomb. The alien must not be destroyed. Eg. it must be visible. The bomb's destroyed flag must be set. In other words, it is alien's first bomb dropping or previous dropped bomb already hit the ground. If these two conditions are fulfilled, the bombing is left to the chance.

if (!b.isDestroyed()) {

b.setY(b.getY() + 1);

if (b.getY() >= GROUND - BOMB\_HEIGHT) {

b.setDestroyed(true);

}

}

If the bomb is not destroyed, it goes 1px to the ground. If it hits the bottom, the destroyed flag is set. The alien is now ready to drop another bomb.

public void keyReleased(KeyEvent e) {

player.keyReleased(e);

}

The actual processing of this particular KeyEvent is delegated to the player sprite.

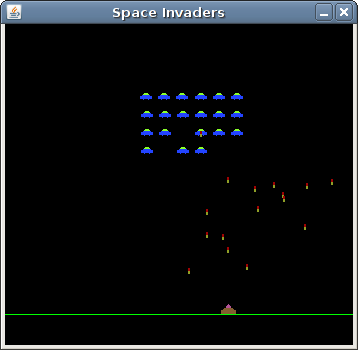


Figure: Space Invaders