```java
package ch.hevs.gdx2d.lunar.main;

import java.util.ArrayList;
import java.util.Random;

import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.graphics.g2d.BitmapFont;
import com.badlogic.gdx.math.Vector2;

import ch.hevs.gdx2d.lib.GdxGraphics;
import ch.hevs.gdx2d.lib.interfaces.DrawableObject;
import ch.hevs.gdx2d.lunar.physics.Constants;
import ch.hevs.gdx2d.lunar.physics.Particles;
import ch.hevs.gdx2d.lunar.physics.PhysicalObject;

public class Spaceship extends PhysicalObject implements DrawableObject {

    private int fuel;
    private int gameNb;

    public boolean thrustUp;
    public boolean thrustLeft;
    public boolean thrustRight;

    private boolean landed;
    private boolean kaputt;
    private boolean firstExplo;

    // Particles stuff
    private ArrayList<Particles> reactor;
    private ArrayList<Particles> explosion;

    static final Random rand = new Random();

    // Textures
    private Texture spaceship;
    private Texture ded;

    public Spaceship(Vector2 p, int gameNb) {
        super(p, new Vector2(0, 0), Constants.BASE_MASS, 50, 50);

        fuel = (int) Constants.MAX_FUEL;
        this.gameNb = gameNb;

        thrustUp = false;
        thrustLeft = false;
        thrustRight = false;

        kaputt = false;
        landed = false;
        firstExplo = true;

        reactor = new ArrayList<Particles>();
        explosion = new ArrayList<Particles>();

        spaceship = new Texture("data/images/ssLandry.png");
        ded = new Texture("data/images/Rip.png");
    }

    @Override
    public void draw(GdxGraphics arg0) {
        // arg0.drawFilledRectangle(position.x, position.y+8, 10, 16, 0, Color.BLUE);
        if (kaputt) {
            Vector2 vec;
            if (firstExplo) {
                for (int i = 0; i < 500; i++) {
                    vec = new Vector2(1, 1).setToRandomDirection();
                    explosion.add(new Particles(new Vector2(position.x, position.y),
                    vec.scl(rand.nextFloat() * 2),
                            rand.nextInt(80),
                            rand.nextBoolean() ? "data/images/fire_particle.png" :
                            "data/images/reactor_particle.png"));
```

```
72                  }
73                  firstExplo = false;
74              }
75              // Explosion animation
76              arg0.draw(ded, position.x - 25, position.y - 25, 50, 50);
77
78              if (explosion.size() != 0) {
79                  for (int i = 0; i < explosion.size(); i++) {
80                      Particles p = explosion.get(i);
81                      p.update();
82                      p.draw(arg0);
83                      if (p.shouldBeDestroyed()) {
84                          explosion.remove(p);
85                      }
86                  }
87              }
88
89          } else {
90              arg0.draw(spaceship, position.x - 25, position.y - 30, 50, 50);
91
92              if (!landed && fuel > 0 && (thrustUp || thrustLeft || thrustRight)) {
93                  // Thrust animation
94                  reactor.add(new Particles(new Vector2(position.x, position.y - 25),
95                          new Vector2(rand.nextFloat() / 2 * (rand.nextBoolean() ? 1 :
                            -1), -2).mulAdd(speed, 0.1f),
96                          rand.nextInt(80),
97                          rand.nextBoolean() ? "data/images/fire_particle.png" :
                            "data/images/reactor_particle.png"));
98              }
99              if (reactor.size() != 0) {
100                 for (int i = 0; i < reactor.size(); i++) {
101                     Particles p = reactor.get(i);
102                     p.update();
103                     p.draw(arg0);
104                     if (p.shouldBeDestroyed()) {
105                         reactor.remove(p);
106                     }
107                 }
108             }
109         }
110
111         drawHUD(arg0);
112     }
113
114     void drawHUD(GdxGraphics arg0) {
115         // Print fond noir
116         arg0.drawFilledRectangle(400, 50, 800, 100, 0, Color.DARK_GRAY);
117         // Print fuel
118         Vector2 POSITION_BAR_FUEL = new Vector2(650, 50);
119         Vector2 POSITION_SPEED = new Vector2(100, 50);
120         Vector2 POSITION_NB_GAME = new Vector2(350, 50);
121
122         arg0.drawRectangle(POSITION_BAR_FUEL.x, POSITION_BAR_FUEL.y, 200, 50, 0);
123         arg0.drawFilledRectangle(POSITION_BAR_FUEL.x - (float) (fuel /
            Constants.MAX_FUEL) * 100 + 100,
124                 POSITION_BAR_FUEL.y, (float) (fuel / Constants.MAX_FUEL) * 200, 50,
                    0, Color.RED);
125         arg0.drawString(POSITION_BAR_FUEL.x - 90, POSITION_BAR_FUEL.y,
126                 "Fuel :" + fuel + "/" + (int) Constants.MAX_FUEL);
127
128         // Print speed
129         BitmapFont bfSpeed = new BitmapFont();
130         bfSpeed.setColor(Color.RED);
131         if (speed.len() < Constants.CRASH_SPEED) {
132             bfSpeed.setColor(Color.GREEN);
133         }
134         arg0.drawString(POSITION_SPEED.x, POSITION_SPEED.y, "Speed :" + (int)
            speed.len() + " m/s", bfSpeed);
135
136         arg0.drawString(POSITION_NB_GAME.x, POSITION_NB_GAME.y, "Apollo " + gameNb);
137
138     }
139
```

```
140         @Override
141         public void step() {
142             // Simulate de thrust from the reactors
143             if (!landed) {
144                 if (thrustUp && fuel > 0) {
145                     force.y = Constants.MAX_THRUST;
146                     fuel--;
147                 } else {
148                     force.y = 0;
149                 }
150 
151                 if (thrustLeft && !thrustRight && fuel > 0) {
152                     force.x = -Constants.MAX_THRUST;
153                     fuel--;
154                 } else if (!thrustLeft && thrustRight && fuel > 0) {
155                     force.x = Constants.MAX_THRUST;
156                     fuel--;
157                 } else {
158                     force.x = 0;
159                 }
160             }
161         }
162 
163         @Override
164         public void removedFromSim() {
165             kaputt = !landed;
166         }
167 
168         @Override
169         public boolean notifyCollision(int energy) {
170             landed = energy < Constants.DESTRUCTION_ENERGY;
171             return (!landed);
172         }
173 
174         public boolean isLanded() {
175             return landed;
176         }
177 
178         public boolean isFinished() {
179             return (kaputt || landed);
180         }
181 
182         public boolean isDry() {
183             return (fuel <= 0);
184         }
185 
186         public boolean isKaputt() {
187             return kaputt;
188         }
189 
190     }
191 
```