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

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

import com.badlogic.gdx.Input;
import com.badlogic.gdx.graphics.Color;
import com.badlogic.gdx.math.Rectangle;
import com.badlogic.gdx.math.Vector2;

import ch.hevs.gdx2d.lib.GdxGraphics;
import ch.hevs.gdx2d.lunar.physics.Constants;
import ch.hevs.gdx2d.lunar.physics.Ground;
import ch.hevs.gdx2d.lunar.physics.Particles;
import ch.hevs.gdx2d.lunar.physics.PhysicsSimulator;
import ch.hevs.gdx2d.components.audio.MusicPlayer;
import ch.hevs.gdx2d.components.audio.SoundSample;
import ch.hevs.gdx2d.desktop.PortableApplication;

public class LunarLander_Main extends PortableApplication {

    // Game components
    PhysicsSimulator physics;
    Spaceship ssLandry;
    Ground sol;
    LandZone lz;
    ArrayList<Gegner> meteors;
    private int gameNb;

    // music
    MusicPlayer music;
    SoundSample noFuel;
    SoundSample bruitExplosion;
    SoundSample winSound;
    SoundSample pew;
    private boolean doSoundFuel = true;
    private boolean doExplosion = true;
    private boolean doWinSound = true;

    // Shooting related
    private ArrayList<Particles> laserExplo;
    boolean mouseActive = false;
    Vector2 positionClick;
    static int waitLaser;

    // Stars particles
    private ArrayList<Particles> stars;
    static final Random rand = new Random();
    static int waitStar;

    public LunarLander_Main() {
        super(Constants.WIN_WIDTH, Constants.WIN_HEIGHT);
    }

    @Override
    public void onInit() {
        setTitle("LunarLandry (Team PLS)");
        gameNb = 1;
        waitStar = 0;
        waitLaser = 0;
        playMusic();
        ssLandry = new Spaceship(new Vector2(100, 700), gameNb);
        sol = new Ground();
        lz = new LandZone(sol.getPolyPoint(Constants.FLAT_ZONE));
        physics = new PhysicsSimulator(Constants.WIN_WIDTH, Constants.WIN_HEIGHT);
        physics.changePlayground(sol.getPolygon(), lz);
        physics.addSimulatableObject(ssLandry);
        stars = new ArrayList<Particles>();
        laserExplo = new ArrayList<Particles>();
        meteors = new ArrayList<Gegner>();
        meteors.add(new Gegner(new Vector2(400, 700)));
        physics.addSimulatableObject(meteors.get(0));
    }
```

```
74
75      @Override
76      public void onGraphicRender(GdxGraphics g) {
77          // Clears the screen
78          g.clear();
79
80          // Simulate every object
81          physics.simulate_step();
82
83          // Draw basic layout
84          // g.drawFPS();
85          // g.drawSchoolLogo();
86
87          // Draw the stars on the background
88          drawStars(g, 200);
89
90          // Spaceship
91          ssLandry.draw(g);
92
93          // Meteors
94          if (meteors.size() != 0) {
95              for (int i = 0; i < meteors.size(); i++) {
96                  meteors.get(i).draw(g);
97              }
98          }
99          drawBoundingBoxes(g);
100
101         if (ssLandry.isFinished() && ssLandry.isKaputt()) {
102             g.drawStringCentered(660, "Appuiez sur 'R' pour recommencer");
103         }
104         if (ssLandry.isFinished() && ssLandry.isLanded()) {
105             g.drawStringCentered(660, "Appuiez sur 'R' pour continuer");
106         }
107         playSound();
108         g.drawFilledPolygon(sol.getPolygon(), Color.LIGHT_GRAY);
109         drawLandZone(g);
110         // g.drawLine(0, Constants.GROUND_ALTITUDE, Constants.WIN_WIDTH,
111         // Constants.GROUND_ALTITUDE, Color.WHITE);
112         drawLaser(g);
113         drawLaserExplo(g, 80);
114
115     }
116
117     void drawBoundingBoxes(GdxGraphics arg0) {
118         if (Constants.DRAW_BOUNDINGBOXES) { // Hitboxes
119             Rectangle box = ssLandry.getBoundingBox();
120             arg0.drawRectangle(box.getX() + box.getWidth() / 2, box.getY() +
                box.getHeight() / 2, box.getWidth(),
121                     box.getHeight(), 0);
122             if (meteors.size() != 0) {
123                 for (int i = 0; i < meteors.size(); i++) {
124                     box = meteors.get(i).getBoundingBox();
125                     arg0.drawRectangle(box.getX() + box.getWidth() / 2, box.getY() +
                        box.getHeight() / 2,
126                             box.getWidth(), box.getHeight(), 0);
127                 }
128             }
129         }
130     }
131
132     void drawLaser(GdxGraphics arg0) {
133         if ((mouseActive || waitLaser > 0) && !ssLandry.isFinished()) {
134             arg0.drawLine(positionClick.x, positionClick.y, ssLandry.position.x,
                ssLandry.position.y, Color.RED);
135             waitLaser--;
136         }
137     }
138
139     void drawLaserExplo(GdxGraphics arg0, int age) {
140         // Laser logik
141         if (mouseActive && !ssLandry.isFinished() && waitLaser < 0) {
142             pew = new SoundSample("data/sons/BruitLaser_low.mp3");
143             pew.play();
```

```
144                 pew.setVolume(0.1f);
145                 mouseActive = false;
146                 waitLaser = 30;
147                 if (meteors.size() != 0) {
148                     for (int i = 0; i < meteors.size(); i++) {
149                         if (meteors.get(i).getBoundingBox().contains(positionClick)) {
150                             physics.removeObjectFromSim(meteors.get(i));
151                         }
152                     }
153                 }
154 
155                 Vector2 vec;
156                 for (int i = 0; i < 100; i++) {
157                     vec = new Vector2(1, 1).setToRandomDirection();
158                     laserExplo.add(new Particles(new Vector2(positionClick.x,
                    positionClick.y), vec.scl(0.2f),
159                             rand.nextInt(age),
160                             rand.nextBoolean() ? "data/images/fire_particle.png" :
                            "data/images/reactor_particle.png"));
161                 }
162             }
163             // Explosion laser animation
164             if (laserExplo.size() != 0) {
165                 for (int j = 0; j < laserExplo.size(); j++) {
166                     Particles p = laserExplo.get(j);
167                     p.update();
168                     p.draw(arg0);
169                     if (p.shouldBeDestroyed()) {
170                         laserExplo.remove(p);
171                     }
172                 }
173             }
174         }
175 
176         void drawLandZone(GdxGraphics arg0) {
177             Color color = Color.RED;
178             if (ssLandry.isLanded()) {
179                 color = Color.GREEN;
180             }
181             arg0.drawFilledRectangle(lz.landBox.getX() + Constants.Z_WIDTH / 2,
            lz.landBox.getY() + Constants.Z_HEIGHT / 2,
182                     Constants.Z_WIDTH, Constants.Z_HEIGHT, 0, color);
183         }
184 
185         void drawStars(GdxGraphics arg0, int age) {
186 
187             waitStar++;
188 
189             // Adds a star every n frames
190             if (waitStar == 5) {
191                 final String img = "data/images/star.png";
192                 final String img2 = "data/images/star2.png";
193                 final String img3 = "data/images/star4big.png";
194                 String imgRand;
195 
196                 int value = (int) (Math.random() * 20);
197                 switch (value) {
198                 case 3:
199                     imgRand = img2;
200                     age = age / 3;
201                     break;
202                 case 6:
203                     imgRand = img3;
204                     age = age / 3;
205                     break;
206                 default:
207                     imgRand = img;
208                     break;
209                 }
210                 stars.add(new Particles(new Vector2(rand.nextFloat() *
                Constants.WIN_WIDTH,
211                         (rand.nextFloat() * (Constants.WIN_WIDTH -
                        Constants.GROUND_ALTITUDE)) + Constants.GROUND_ALTITUDE),
```

```
                    new Vector2(0.1f, 0), age, imgRand));

            waitStar = 0;
        }

        // Draw the stars
        if (stars.size() != 0) {
            for (int i = 0; i < stars.size(); i++) {
                Particles p = stars.get(i);
                p.update();
                p.draw(arg0);
                if (p.shouldBeDestroyed()) {
                    stars.remove(p);
                }
            }
        }
    }

    public int getNbGame() {
        return gameNb;
    }

    void playMusic() {
        if (rand.nextInt(100) <= 10) {
            music = new MusicPlayer("data/sons/zambla.mp3");
        } else {
            music = new MusicPlayer("data/sons/sound1_low.mp3");
        }
        music.loop();
    }

    void playSound() {
        if (ssLandry.isDry() && doSoundFuel) {
            final String dry1 = "data/sons/NoFuel.mp3";
            final String dry2 = "data/sons/Ecolo.mp3";
            final String dry3 = "data/sons/Sub.mp3";
            String dry;
            int value = rand.nextInt(4);
            switch (value) {
            case 2:
                dry = dry2;
                break;
            case 3:
                dry = dry3;
                break;
            default:
                dry = dry1;
                break;
            }
            noFuel = new SoundSample(dry);
            noFuel.play();
            doSoundFuel = false;
        }
        if (ssLandry.isKaputt() && doExplosion) {
            gameNb = 1;
            final String kaputt1 = "data/sons/bruitExplo.mp3";
            final String kaputt2 = "data/sons/doucement.mp3";
            String kaputt;
            int value = rand.nextInt(3);
            switch (value) {
            case 2:
                kaputt = kaputt2;
                break;
            default:
                kaputt = kaputt1;
                break;
            }
            bruitExplosion = new SoundSample(kaputt);
            bruitExplosion.play();
            doExplosion = false;
        }
        if (ssLandry.isLanded() && doWinSound) {
            if (gameNb == 11) {
```

```
                winSound = new SoundSample("data/sons/OneSmallStep.mp3");
            } else if (gameNb < 11) {
                winSound = new SoundSample(rand.nextBoolean() ?
                "data/sons/Sympa.mp3" : "data/sons/bof_low.mp3");
            } else {
                winSound = new SoundSample("data/sons/VSS.mp3");
            }
            winSound.play();
            winSound.mofidyPlayingVolument(0.1f, 1);
            doWinSound = false;
            gameNb++;
        }
    }

    @Override
    public void onClick(int x, int y, int button) {
        super.onClick(x, y, button);
        mouseActive = true;
        if (waitLaser <= 0) {
            positionClick = new Vector2(x, y);
        }
    }

    @Override
    public void onRelease(int x, int y, int button) {
        super.onRelease(x, y, button);
        if (waitLaser <= 0) {
            positionClick.x = x;
            positionClick.y = y;
        }
        mouseActive = false;
    }

    @Override
    public void onKeyUp(int keycode) {
        switch (keycode) {
        case Input.Keys.UP:
            ssLandry.thrustUp = false;
            break;
        case Input.Keys.LEFT:
            ssLandry.thrustLeft = false;
            break;
        case Input.Keys.RIGHT:
            ssLandry.thrustRight = false;
            break;
        case Input.Keys.W:
            ssLandry.thrustUp = false;
            break;
        case Input.Keys.A:
            ssLandry.thrustLeft = false;
            break;
        case Input.Keys.D:
            ssLandry.thrustRight = false;
            break;
        default:
            break;
        }
    }

    @Override
    public void onKeyDown(int keycode) {
        switch (keycode) {
        case Input.Keys.UP:
            ssLandry.thrustUp = true;
            break;
        case Input.Keys.LEFT:
            ssLandry.thrustLeft = true;
            break;
        case Input.Keys.RIGHT:
            ssLandry.thrustRight = true;
            break;
        case Input.Keys.W:
            ssLandry.thrustUp = true;
```

```
357                 break;
358             case Input.Keys.A:
359                 ssLandry.thrustLeft = true;
360                 break;
361             case Input.Keys.D:
362                 ssLandry.thrustRight = true;
363                 break;
364             case Input.Keys.R:
365                 if (ssLandry.isFinished()) {
366                     replay();
367                 }
368             default:
369                 break;
370             }
371         }
372 
373         public void replay() {
374             if (ssLandry.isLanded()) {
375                 winSound.stop();
376             }
377 
378             physics.removeAllObjectsfromSim();
379             ssLandry = new Spaceship(new Vector2(100, 700), gameNb);
380             sol = new Ground();
381             lz = new LandZone(sol.getPolyPoint(Constants.FLAT_ZONE));
382             physics.changePlayground(sol.getPolygon(), lz);
383             physics.addSimulatableObject(ssLandry);
384 
385             meteors.clear();
386             for (int i = 0; i < gameNb; i++) {
387                 meteors.add(new Gegner(new Vector2(rand.nextInt(300) + 400,
                    rand.nextInt(300) + 500)));
388                 physics.addSimulatableObject(meteors.get(i));
389             }
390 
391             doSoundFuel = true;
392             doExplosion = true;
393             doWinSound = true;
394         }
395 
396         public static void main(String[] args) {
397             new LunarLander_Main();
398         }
399     }
400 
```