import java.awt.Polygon;

import java.awt.Rectangle;

import java.util.ArrayList;

import javax.swing.JOptionPane;

/\*\*

\*

\*Description This class generates asteriods. The player specifies how many asteroids there are. Asteroids interact with lasers by getting destroyed by them, and they also destroy ships.

@Author Eric Feng

@Version 5.26.2020

\*

\*/

public class AsteroidObstacle extends Polygon {

// Variables for positions of objects

private int obstacleXPos;

private int obstacleYPos;

private int xMove = 2;

private int yMove = 2;

private static int destroyedObstacles = 0;

private static int timesExploded = 0;

boolean obstacleOnScreen = true;

// Get the width and height of the Game

private int heightOfWindow = AsteroidGame.windowHeight;

private int widthOfWindow = AsteroidGame.windowWidth;

// Hold the positions of the Obstacle

private int widthOfObstacle = 29;

private int heightOfObstacle = 31;

// Initial posisitions of obstacle

private static int[] sObstacleXArray = { 30, 30, 30, 30, 30, 30, 30, 30, 0, 0, 0, 0, 0 };

private static int[] sObstacleYArray = { 0, 0, 0, 0, 30, 30, 30, 30, 30, 30, 30, 0, 0 };

public static ArrayList<AsteroidObstacle> obstacles;

/\*\*

\*Description This function generates the starting positions of the various different asteriods, depending on the amount of asteroids the player added. It is done with a random number generator.

\* @param obstacleXArray Creates an array for all the X coordinates

\* @param obstacleYArray Creates an array for all the Y coordinates

\* @param pointsInPoly Creates the points for polygons

\* @param randomSTartXPos Generates the random starting X coordinates

\*

\*/

public AsteroidObstacle(int[] obstacleXArray, int[] obstacleYArray, int pointsInPoly, int randomStartXPos,

int randomStartYPos) {

super(obstacleXArray, obstacleYArray, pointsInPoly);

// Set the x and y direction to a random integer

int randomX = (int) (Math.random() \* 8);

int randomY = (int) (Math.random() \* 8);

if (randomX != 4 && randomY != 4) {

this.xMove = randomX - 4;

this.yMove = randomY - 4;

}

}

// Make the setter and getter methods

/\*\*

\* Description This functions gets an array of X coordinates of the asteroids, which are randomly generated. The length of the array depends on how many asteroids the user intends to play with in the game.

\* @return sObstacleXArray Records the X coordinates in an array

\*

\*/

public static int[] getStartingObstacleXArray() {

return sObstacleXArray;

}

/\*\*

\* Desription This functions gets an array of Y coordinates of the asteroids, which are randomly generated. The length of the array depends on how many asteroids the user intends to play with in the game.

\* @return sObstacleYArray Records the Y coordinates in an array

\*/

public static int[] getStartingObstacleYArray() {

return sObstacleYArray;

}

public static void resetdestroyedObstacles() {

AsteroidObstacle.destroyedObstacles = 0;

}

public static void resetTimesExploded() {

AsteroidObstacle.timesExploded = 0;

}

// Collosion detection of Obstacles

public Rectangle getBounds() {

return new Rectangle(super.xpoints[0], super.ypoints[0], widthOfObstacle, heightOfObstacle);

}

/\*\*

\* Description: This function moves the asteroids in the game by updating their position and considering the factors of spaceships, lasers, their starting position, and their velocity. It applies their velocity and center to their current x and y coordinates and updates it.

\* @param obstacleXArray The array of X-coordinates for the asteriods.

\* @param obstacleYArray The array of Y-coordinates for the asteriods.

\* @param pointsInPoly The points in the polygons that make up the asteroids

\* @param randomStartXPos The random x positions that the asteroids start at

\* @param randomStartYPos The random y positions that the asteroids begin at

\* @param SpaceShip The properties of a spaceship

\* @param Lasers The properties of a laser

\* @return obstacle.Xarray

\* @return obstacle.Yarray

\* @return Xmove

\* @return Ymove

\*/

// Used to move to obstacles and check for collisions between them

public void move(SpaceShip SpaceShip, ArrayList<LaserShoot> lasers) {

// Create a new rectangle based on the current coordinates

Rectangle obstacleToCheck = this.getBounds();

for (AsteroidObstacle obstacle : obstacles) {

if (obstacle.obstacleOnScreen) {

Rectangle otherobstacle = obstacle.getBounds();

// Checking to see if Obstacles intersect with each other

if (obstacle != this && otherobstacle.intersects(obstacleToCheck)) {

// Checking both coordinates of Obstacle

int tempXMove = this.xMove;

int tempYMove = this.yMove;

this.xMove = obstacle.xMove;

this.yMove = obstacle.yMove;

obstacle.xMove = tempXMove;

obstacle.yMove = tempYMove;

}

// If the obstacle's collision box intersects with ship collision box

Rectangle shipBox = SpaceShip.getBounds();

if (SpaceShip.interaction == true) {

if (otherobstacle.intersects(shipBox)) {

// When dead, respawn ship

SpaceShip.interaction = false;

SpaceShip.setXShip(SpaceShip.heightOfWindow / 2);

SpaceShip.setYShip(SpaceShip.heightOfWindow / 2);

SpaceShip.setXVelocity(0);

SpaceShip.setYVelocity(0);

AsteroidGame.score -= 10;

AsteroidGame.pointsLabel.setText("Score: " + AsteroidGame.firstName + " "

+ AsteroidGame.lastName + " : " + AsteroidGame.score);

System.out.println("HIT! You lose 10 points.");

System.out.println("Score: " + AsteroidGame.firstName + " " + AsteroidGame.lastName + " : "

+ AsteroidGame.score);

timesExploded += 1;

}

}

// Does ship have contact with laser

for (LaserShoot laser : lasers) {

if (laser.laserOnScreen) {

if (otherobstacle.contains(laser.getlaserX1(), laser.getlaserY1())

| obstacleToCheck.contains(laser.getlaserX1(), laser.getlaserY1())) {

// Obstacles disappear

obstacle.obstacleOnScreen = false;

laser.laserOnScreen = false;

AsteroidGame.score += 10;

AsteroidGame.pointsLabel.setText("Score: " + AsteroidGame.firstName + " "

+ AsteroidGame.lastName + " : " + AsteroidGame.score);

System.out.println("You destroyed an obstacle! You gain 10 points.");

System.out.println("Score: " + AsteroidGame.firstName + " " + AsteroidGame.lastName + " : "

+ AsteroidGame.score);

destroyedObstacles += 1;

System.out.println("Obstacles destroyed: " + destroyedObstacles);

System.out.println("total obstacles: " + AsteroidGame.numberOfAsteroids);

// If all obstacles are destroyed, end game

if (destroyedObstacles >= AsteroidGame.numberOfAsteroids) {

System.out.println("Game Over!");

AsteroidGame.pointsLabel.setText("Game Over! Final Score: " + AsteroidGame.score);

AsteroidGame.displayResults(destroyedObstacles, timesExploded);

}

}

}

}

}

}

obstacleXPos = super.xpoints[0];

obstacleYPos = super.ypoints[0];

// When obstacles hit a wall it flies back to center

if (obstacleXPos < 0 || (obstacleXPos + 29) > widthOfWindow)

this.xMove \*= -1;

if (obstacleYPos < 0 || (obstacleYPos + 31) > heightOfWindow)

this.yMove \*= -1;

// Move the values of the points for the polygon to the center

for (int i = 0; i < super.xpoints.length; i++) {

super.xpoints[i] += xMove;

super.ypoints[i] += yMove;

}

}

// Used to create array based on random x positions

/\*\*

\* Description This function takes in the new x positions of the asteroids and stores them in a temporary array for them to be updated later.

\* @return tempObstacleArray

\*/

public static int[] getObstacleXArray(int randomStartXPos) {

// Creates new array

int[] tempObstacleXArray = (int[]) sObstacleXArray.clone();

for (int i = 0; i < tempObstacleXArray.length; i++) {

tempObstacleXArray[i] += randomStartXPos;

}

return tempObstacleXArray;

}

/\*\*

\* Description This function takes in the new y positions of the asteroids and stores them in a temporary array for them to be updated later.

\* @return temp ObstacleYArray

\*/

public static int[] getObstacleYArray(int randomStartYPos) {

int[] tempObstacleYArray = (int[]) sObstacleYArray.clone();

for (int i = 0; i < tempObstacleYArray.length; i++) {

tempObstacleYArray[i] += randomStartYPos;

}

return tempObstacleYArray;

}

}