班级　NZ222　姓名　霍梓航　学号　225858

Labs 5

```java
package com.hamhuo.massey;

import java.awt.*;
import java.awt.event.*;

public class Lab5 extends GameEngine {
    // Main Function
    public static void main(String args[]) {
        // Warning: Only call createGame in this function
        // Create a new Lab5
        createGame(new Lab5());
    }

    //-------------------------------------------------------
    // Spaceship
    //-------------------------------------------------------

    // Image of the spaceship
    Image spaceshipImage;

    // Spaceship position
    double spaceshipPositionX;
    double spaceshipPositionY;

    // Spaceship velocity
    double spaceshipVelocityX;
    double spaceshipVelocityY;

    // Spaceship angle
    double spaceshipAngle;

    // Init Spaceship Function
    public void initSpaceship() {
        // Load the spaceship sprite
        spaceshipImage = subImage(spritesheet, 0, 0, 240, 240);

        // Setup Spaceship variables
        spaceshipPositionX = width() / 2;
        spaceshipPositionY = height() / 2;
        spaceshipVelocityX = 0;
        spaceshipVelocityY = 0;
        spaceshipAngle = 0;
    }

    // Function to draw the spaceship
    public void drawSpaceship() {
        // Save the current transform
        saveCurrentTransform();

        // Translate to the position of the asteroid
        translate(spaceshipPositionX, spaceshipPositionY);

        // Rotate the drawing context around the angle of the
        rotate(spaceshipAngle);

        // Draw the actual spaceship
        drawImage(spaceshipImage, -30, -30, 60, 60);

        // Restore last transform to undo the rotate and translate transforms
        restoreLastTransform();
    }

    // Code to update 'move' the spaceship
    public void updateSpaceship(double dt) {
        if (up == true) {
            // Increase the velocity of the spaceship
            // as determined by the angle
            spaceshipVelocityX += sin(spaceshipAngle) * 250 * dt;
            spaceshipVelocityY -= cos(spaceshipAngle) * 250 * dt;
        }

        // If the user is holding down the left arrow key
        if (left == true) {
            // Make the spaceship rotate anti-clockwise
            spaceshipAngle -= 250 * dt;
        }

        // If the user is holding down the right arrow key
        if (right == true) {
            // Make the spaceship rotate clockwise
            spaceshipAngle += 250 * dt;
        }

        // Make the spaceship move forward
        spaceshipPositionX += spaceshipVelocityX * dt;
        spaceshipPositionY += spaceshipVelocityY * dt;

        // If the spaceship reaches the right edge of the screen
        // 'Warp' it back to the left edge
        if (spaceshipPositionX > width()) {
            spaceshipPositionX -= width();
        }

        // If the spaceship reaches the left edge of the screen
        // 'Warp' it back to the right edge
        if (spaceshipPositionX < 0) {
            spaceshipPositionX += width();
        }

        // If the spaceship reaches the top edge of the screen
        // 'Warp' it back to the bottom edge
        if (spaceshipPositionY > height()) {
            spaceshipPositionY -= height();
        }

        // If the spaceship reaches the bottom edge of the screen
        // 'Warp' it back to the top edge
        if (spaceshipPositionY < 0) {
            spaceshipPositionY += height();
        }
    }


    //-------------------------------------------------------
    // Laser
    //-------------------------------------------------------

    // Image of the laser
    Image laserImage;

    //  // Laser position
//  double laserPositionX;
//  double laserPositionY;
//
//  // Laser velocity
//  double laserVelocityX;
//  double laserVelocityY;
//
//  // Laser Angle
//  double laserAngle;
```

```java
//
//  // Laser active
//  boolean laserActive;
    int maxLasers = 5;
    double[] laserPositionX = new double[maxLasers];
    double[] laserPositionY = new double[maxLasers];
    double[] laserVelocityX = new double[maxLasers];
    double[] laserVelocityY = new double[maxLasers];
    double[] laserAngle = new double[maxLasers];
    boolean[] laserActive = new boolean[maxLasers];

    // Initialise Laser
    public void initLaser() {
        laserImage = subImage(spritesheet, 240, 0, 240, 240);
        // Make laser inactive
        for (int i = 0; i < maxLasers; i++) {
            laserActive[i] = false;
        }
    }

    // Function to shoot a new laser
    public void fireLaser() {
        // Can only fire a laser if there isn't already one active
//      if(laserActive == false) {
//          // Set the laser position as the current spaceship position
//          laserPositionX = spaceshipPositionX;
//          laserPositionY = spaceshipPositionY;
//
//          // And make it move in the same direction as the spaceship is facing
//          laserVelocityX =  sin(spaceshipAngle) * 250;
//          laserVelocityY = -cos(spaceshipAngle) * 250;
//
//          // And face the same direction as the spaceship
//          laserAngle = spaceshipAngle;
//
//          // Set it to active
//          laserActive = true;
//      }
        for (int i = 0; i < maxLasers; i++) {
            if (!laserActive[i]) {
                laserPositionX[i] = spaceshipPositionX;
                laserPositionY[i] = spaceshipPositionY;
                laserVelocityX[i] = sin(spaceshipAngle) * 250;
                laserVelocityY[i] = -cos(spaceshipAngle) * 250;
                laserAngle[i] = spaceshipAngle;
                laserActive[i] = true;
                break;
            }
        }
    }

    // Function to draw the laser
    public void drawLaser() {
//      // Only draw the laser if it's active
//      if(laserActive) {
//          // Save the current transform
//          saveCurrentTransform();
//
//          // ranslate to the position of the laser
//          translate(laserPositionX, laserPositionY);
//
//          // Rotate the drawing context around the angle of the laser
//          rotate(laserAngle);
//
//          // Draw the actual laser
////            drawLine(0, -10, 0, 10);
//          drawImage(laserImage, -30,-30,60,60);
//          // Restore last transform to undo the rotate and translate transforms
//          restoreLastTransform();
//      }
        for (int i = 0; i < maxLasers; i++) {
            if (laserActive[i]) {
                // 保存当前变换
                 saveCurrentTransform();

                // 转换到激光的位置
                 translate(laserPositionX[i], laserPositionY[i]);

                // 绕激光的角度旋转
                 rotate(laserAngle[i]);

                // 绘制激光
                 drawImage(laserImage, -30, -30, 60, 60);

                // 恢复变换
                 restoreLastTransform();
            }
        }
    }

    // Function to update 'move' the laser
    public void updateLaser(double dt) {
```

```
//          // Move the Laser
//          laserPositionX += laserVelocityX * dt;
//          laserPositionY += laserVelocityY * dt;
//
//          // If the laser reaches the left edge of the screen
//          // Destroy the laser
//          if (laserPositionX < 0) {
//              laserActive = false;
//          }
//
//          // If the laser reaches the right edge of the screen
//          // Destroy the laser
//          if (laserPositionX >= width()) {
//              laserActive = false;
//          }
//
//          // If the laser reaches the top edge of the screen
//          // Destroy the laser
//          if (laserPositionY < 0) {
//              laserActive = false;
//          }
//
//          // If the laser reaches the bottom edge of the screen
//          // Destroy the laser
//          if (laserPositionY >= height()) {
//              laserActive = false;
//          }
        for (int i = 0; i < maxLasers; i++) {
            if (laserActive[i]) {
                // 更新激光的位置
                 laserPositionX[i] += laserVelocityX[i] * dt;
                laserPositionY[i] += laserVelocityY[i] * dt;

                // 如果激光超出屏幕，停止它
                 if (laserPositionX[i] < 0 || laserPositionX[i] >= width() ||
                        laserPositionY[i] < 0 || laserPositionY[i] >= height()) {
                    laserActive[i] = false;
                }
            }
        }
    }

    //--------------------------------------------------------
    // Asteroid
    //--------------------------------------------------------
```

```
// Image of the asteroid
Image asteroidImage;

// Asteroid Position
double asteroidPositionX;
double asteroidPositionY;

double asteroidVelocityX;
double asteroidVelocityY;

double asteroidAngle;

double asteroidRadius;

public void randomAsteroid() {
    asteroidImage = subImage(spritesheet, 0, 480, 240, 240);
    // Random position
    asteroidPositionX = rand(width());
    asteroidPositionY = rand(height());

    // Random Velocity
    asteroidVelocityX = -50 + rand(100);
    asteroidVelocityY = -50 + rand(100);

    // Random Angle
    asteroidAngle = rand(360);

    // Fixed Radius
    asteroidRadius = 30;
}

// Function to update 'move' the asteroid
public void updateAsteroid(double dt) {
    // Move the asteroid
    asteroidPositionX += asteroidVelocityX * dt;
    asteroidPositionY += asteroidVelocityY * dt;

    // If the asteroid reaches the left edge of the screen
    // 'Warp' it back to the other side of the screen
    if (asteroidPositionX < 0) {
        asteroidPositionX += width();
    }

    // If the asteroid reaches the right edge of the screen
    // 'Warp' it back to the other side of the screen
    if (asteroidPositionX >= width()) {
            asteroidPositionX -= width();
        }

        // If the asteroid reaches the top edge of the screen
        // 'Warp' it back to the other side of the screen
        if (asteroidPositionY < 0) {
            asteroidPositionY += height();
        }

        // If the asteroid reaches the bottom edge of the screen
        // 'Warp' it back to the other side of the screen
        if (asteroidPositionY >= height()) {
            asteroidPositionY -= height();
        }
    }

    // Function to draw the asteroid
    public void drawAsteroid() {
        // Save the current transform
        saveCurrentTransform();

        // ranslate to the position of the asteroid
        translate(asteroidPositionX, asteroidPositionY);

        // Rotate the drawing context around the angle of the asteroid
        rotate(asteroidAngle);

        // Draw the actual asteroid
//      drawCircle(0, 0, asteroidRadius);
        drawImage(asteroidImage, -30, -30, 60, 60);

        // Restore last transform to undo the rotate and translate transforms
        restoreLastTransform();
    }

    //-------------------------------------------------------
    // Game
    //-------------------------------------------------------

    // Spritesheet
    Image spritesheet;

    // Keep track of keys
    boolean left, right, up, down;
    boolean gameOver;
    // Function to initialise the game
    public void init() {
        // Load sprites
        spritesheet                                                                                   =
loadImage("C:\\Users\\HuoZihang\\Documents\\GitHub\\StudentSphere\\Massey\\159.261\\Lab5\\sr
c\\main\\resources\\spritesheet.png");

        // Setup booleans
        left = false;
        right = false;
        up = false;
        down = false;

        gameOver = false;

        // Initialise Spaceship
        initSpaceship();

        // Setup Laser
        initLaser();

        //setup rocket engine
        initRocket();

        // Setup Asteroid
        randomAsteroid();
    }

    // Updates the display
    public void update(double dt) {
        // If the game is over
        if (gameOver == true) {
            // Don't try to update anything.
            return;
        }

        // Update the spaceship
        updateSpaceship(dt);

        // Update the laser
        updateLaser(dt);

        // Update Asteroid
        updateAsteroid(dt);

        // Detect Collision between Laser and Asteroid
//        if (laserActive == true) {
//                        if  (distance(laserPositionX,  laserPositionY,  asteroidPositionX,
asteroidPositionY) < asteroidRadius * 1.2) {
//                // Destroy the laser
//                laserActive = false;
//                // Create a new random Asteroid
//                randomAsteroid();
//            }
//        }
        for (int i = 0; i < maxLasers; i++) {
            if (laserActive[i]) {
                if   (distance(laserPositionX[i],   laserPositionY[i],   asteroidPositionX,
asteroidPositionY) < asteroidRadius * 1.2) {
                    // 激光与小行星碰撞，销毁激光并创建新小行星
                     laserActive[i] = false;
                    randomAsteroid();
                }
            }
        }

        // Detect Collision between Spaceship and Asteroid
        if     (distance(spaceshipPositionX,     spaceshipPositionY,     asteroidPositionX,
asteroidPositionY) < asteroidRadius + 30) {
            // Collision!
            gameOver = true;
        }
    }


    //------------------------------------------------------
    // Rocket Engine
    //------------------------------------------------------

    // Image of the rocketEngine
    Image toLeftImage;
    Image toRightImage;
    Image toUpImage;

    // Init Spaceship Function
    public void initRocket() {
        // Load the spaceship sprite
        toUpImage = subImage(spritesheet, 0, 240, 240, 240);
        toLeftImage = subImage(spritesheet, 240, 240, 240, 240);
        toRightImage = subImage(spritesheet, 480, 240, 240, 240);
    }
```

```
// Function to draw the spaceship
public void drawRocket() {
    // Save the current transform
    saveCurrentTransform();

    // Translate to the position of the asteroid
    translate(spaceshipPositionX, spaceshipPositionY);

    // Rotate the drawing context around the angle of the asteroid
    rotate(spaceshipAngle);

    // Draw the actual spaceship
    if (up == true) {
        drawImage(toUpImage, -30, -30, 60, 60);
    }
    if (right == true) {
        drawImage(toRightImage, -30, -30, 60, 60);
    }
    if (left == true) {
        drawImage(toLeftImage, -30, -30, 60, 60);
    }

    // Restore last transform to undo the rotate and translate transforms
    restoreLastTransform();
}

// This gets called any time the Operating System
// tells the program to paint itself
public void paintComponent() {
    // Clear the background to black
    changeBackgroundColor(black);
    clearBackground(width(), height());

    // If the game is not over yet
    if (gameOver == false) {
        // Draw the Asteroid
        changeColor(white);
        drawAsteroid();

        // Draw the laser (if it's active)
        changeColor(white);
        drawLaser();

        // Draw the Spaceship
        drawSpaceship();
        // Draw the rocket
```

```
            drawRocket();
        } else {
            // If the game is over
            // Display GameOver text
            changeColor(white);
            drawText(width() / 2 - 165, height() / 2, "GAME OVER!", "Arial", 50);
        }
    }

    // Called whenever a key is pressed
    public void keyPressed(KeyEvent e) {
        //T he user pressed left arrow
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            // Record it
            left = true;
        }
        // The user pressed right arrow
        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            // Record it
            right = true;
        }
        // The user pressed up arrow
        if (e.getKeyCode() == KeyEvent.VK_UP) {
            // Record it
            up = true;
        }
        // The user pressed space bar
        if (e.getKeyCode() == KeyEvent.VK_SPACE) {
            // Fire the laser
            fireLaser();
        }
    }

    // Called whenever a key is released
    public void keyReleased(KeyEvent e) {
        // The user released left arrow
        if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            // Record it
            left = false;
        }
        // The user released right arrow
        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            // Record it
            right = false;
        }
        // The user released up arrow
        if (e.getKeyCode() == KeyEvent.VK_UP) {
            // Record it
            up = false;
        }
    }
}
```