

تمرین اول – انجام یک پروژه مقدماتی

عنوان پروژه

شبیه سازی آتش بازی (Fireworks)

شرح کوتاه پروژه

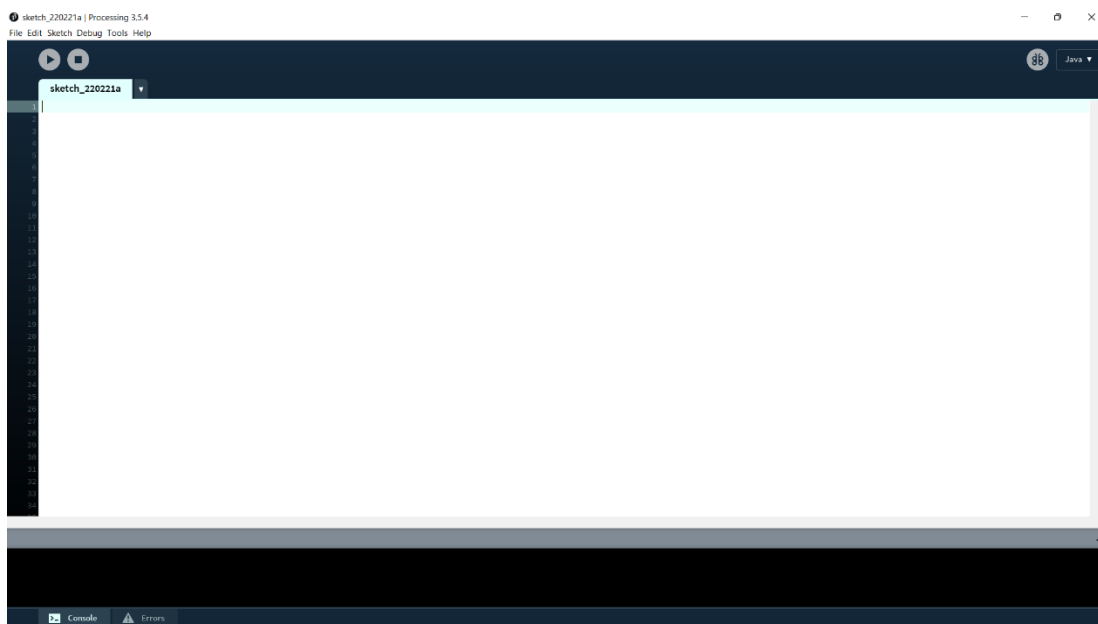
طراحی انیمیشن و شبیه سازی آتش بازی با قابلیت تعامل با کاربر توسط Processing

شرح دقیق تر پروژه

هدف از این پروژه طراحی برنامه ای است برای شبیه سازی آتش بازی در محیط نرم افزار که با کمک موتور Processing ساخته و رندر شده است. در این برنامه به صورت تصادفی آتش بازی هایی در طول زمان فعال شده. همچنین کاربر میتواند با کلیک کردن بر روی صفحه خود آتش بازی را فعال کند. همچنین با فشردن دو دکمه 1 و 2، دو حالت مختلف برای شلیک همزمان تعداد متعدد آتش بازی امکان پذیر است.

مراحل انجام پروژه:

1. نصب فریم ورک و محیط Processing برای نوشتن کد ها و اجرای آنها
این محیط تغییر یافته برای افزایش قدرت رندر کردن جاوا استفاده میشود.



2. آموزش زبان Processing از طریق ویدئو YouTube

پلی لیست Coding Train ویدئو های متعددی برای آموزش این زبان و منطق آن داشته که بسیار مفید بودند.

تمرین اول – انجام یک پروژه مقدماتی

8.6: Pass by Value vs. Pass by Reference - Processing Tutorial

194,266 views • Jul 24, 2015

3.1K DISLIKE SHARE DOWNLOAD THANKS CLIP SAVE

Learning Processing
Gabe Duarte - 38 / 65

3. ادامه آموزش پیشرفته تر Processing از طریق Documentations این زبان
برای مطالعه پیشتر و پیشرفته تر این زبان و قابلیت های آن، به داکيومنتيشن های آن رجوع شد.

Processing

Language Libraries Tools Environment

This reference is for Processing 3.0+. If you have a previous version, use the reference included with your software in the Help menu. If you see any errors or have suggestions, please let us know. If you prefer a more technical reference, visit the Processing Core Javadoc and Libraries Javadoc.

Name ellipseMode()

Examples

Description

Modifies the location from which ellipses are drawn by changing the way in which parameters given to ellipse() are interpreted.

The default mode is ellipseMode(CENTER), which interprets the first two parameters of ellipse() as the shape's center point, while the third and fourth parameters are its width and height.

ellipseMode(RADIUS) also uses the first two parameters of ellipse() as the shape's center point, but uses the third and fourth parameters to specify half of the shape's width and

4. نوشتن نسخه کد سبک برای پیاده سازی اولیه و ارزیابی مفاهیم و منطق این شبیه سازی

این کد صرفاً برای تست کردن امکان اجرای پروژه و توانایی برای پیاده سازی آن بود.

5. بهینه سازی کد و تبدیل به سبک OOP برای مراحل بعد

برای توسعه و وسعت دادن پروژه و استفاده متعدد از توابع و کدهای آن به سبک Object Oriented تغییر یافت.

تمرین اول – انجام یک پروژه مقدماتی

```
class Particle {
    PVector location;
    PVector velocity;
    PVector acceleration;
    float lifespan;
    float spanStep;

    boolean seed = false;

    float hu;

    Particle(float x, float y, float h) {
        hu = h;

        acceleration = new PVector(0, 0);
        velocity = new PVector(0, random(-24, -5));
        location = new PVector(x, y);
        seed = true;
        lifespan = 255.0;
        spanStep = random(1, 10);
    }

    Particle(PVector l, float h) {
        hu = h;
        acceleration = new PVector(0, 0);
        velocity = PVector.random2D();
        velocity.mult(random(30));
        location = l.copy();
        lifespan = 255.0;
        spanStep = random(1, 10);
    }

    void applyForce(PVector force) {
        acceleration.add(force);
    }
}
```

6. اضافه کردن قابلیت های تعاملی توسط کاربرد

کاربر میتواند با فشردن دکمه های 1 یا 2 و با کلیک کردن تغییرات و رویداد های جدید را رغم بزند

تمرین اول – انجام یک پروژه مقدماتی

```
// Costume maneuvers
void keyPressed() {

    fireWorksCount = (int)random(50, 60); // Keep low for slower PCs

    if (key == '1' && coolDown > 300) {
        X = 0;
        Y = height;

        for (int i = 0; i < fireWorksCount; i++)
            fireworks.add(new Firework(new PVector(X += (width / (float)fireWorksCount), Y)));

        coolDown = 0;
    } else if (key == '2' && coolDown > 300) {
        Y = height;
        maneuver = true;
        coolDown = 0;

        if ((int)random(2) == 0) {
            dir = 1;
            X = 0;
        } else {
            dir = -1;
            X = width;
        }
    } else if (key == ' ') {
        if (debug)
            debug = false;
        else
            debug = true;
    } else if (keyCode == UP)
        chance += 0.001;
    else if (keyCode == DOWN)
        chance -= 0.001;
}
```

7. طراحی منطق Pooling برای نمایش و رندر کردن تعداد بیشتری از Object ها

```
class Firework {

    ArrayList<Particle> particles; // An arraylist for all the particles
    Particle firework;
    float hu;

    Firework() {
        hu = random(255);
        firework = new Particle(random(-100, width + 100), height, hu);
        particles = new ArrayList<Particle>(); // Initialize the arraylist
    }

    Firework(PVector location) {
        hu = random(255);
        firework = new Particle(location.x, location.y, hu);
        particles = new ArrayList<Particle>(); // Initialize the arraylist
    }

    boolean done() {
        if (firework == null && particles.isEmpty()) {
            return true;
        } else {
            return false;
        }
    }

    void run() {
        if (firework != null) {
            fill(hu, 255, 255);
            firework.applyForce(gravity);
            firework.update();
            firework.display();

            if (firework.explode()) {
                // ...
            }
        }
    }
}
```

8. کامل کردن طراحی و بهبود موتور رندر کردن برای شبیه سازی تمیز و سریع تر

برای بهبود کیفیت تمام اسکرین شات های داخل فایل، ضمیمه شده اند. همچنین سورس کد مربوط به شبیه سازی هم ضمیمه شده است.