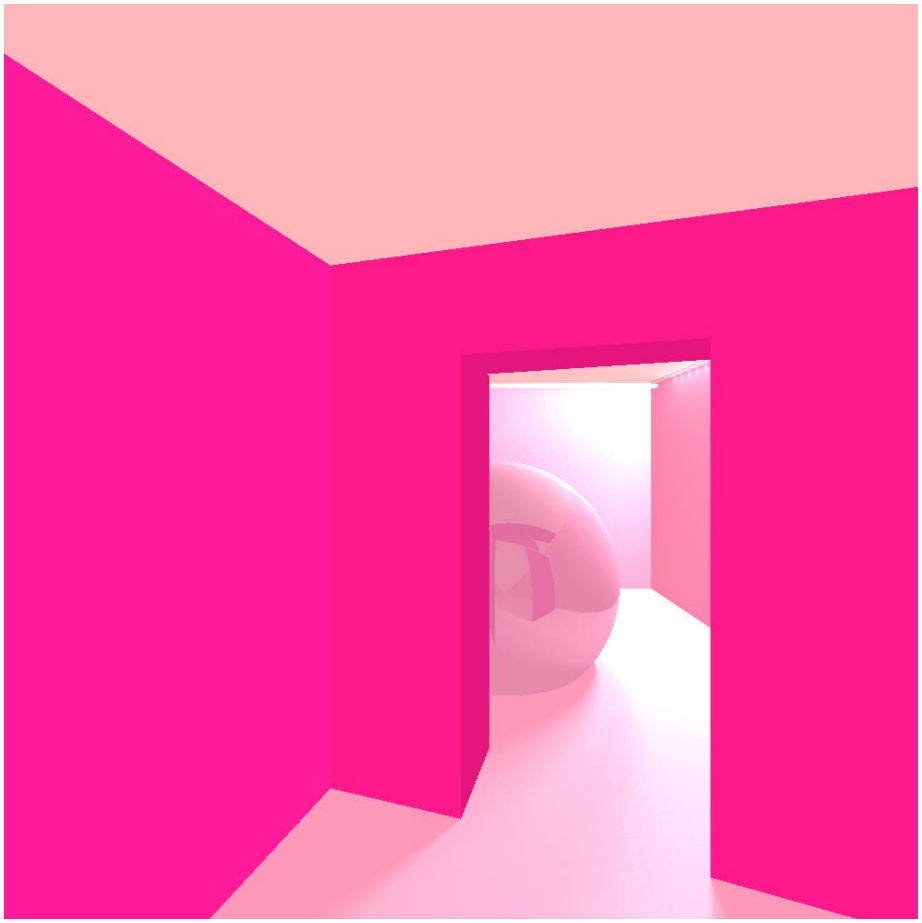
27.06.23 בס"ד

Introduction to Software Engineering

Mini Project the Pink Room With A Led– Report



השותפות לתהליך:
פזית אקבשב, לאה שטרנבוך
שיינא כורם, מיכל סלוצקין

מרצה מלווה:
מר אליעזר גינסבורג

בשלב מיני פרויקט 1 נדרשנו ליצור סצנה שמכילה בתוכה מעל 10 אובייקטים ו3 מקורות אור.
בסצנה שלנו יצרנו חדר אשר במרכזו עומדת ספרה , בנקודת החיתוך בין הקירות לתקרת החדר יצרנו לדים שהם מקורות האור שלנו בנוסף יצרנו לחדר מראה תלת ממדי על ידי כך שהוספנו בקדמת החדר קיר ומשקוף שיצרים דלת לחדר שבו נמצאת הספרה.

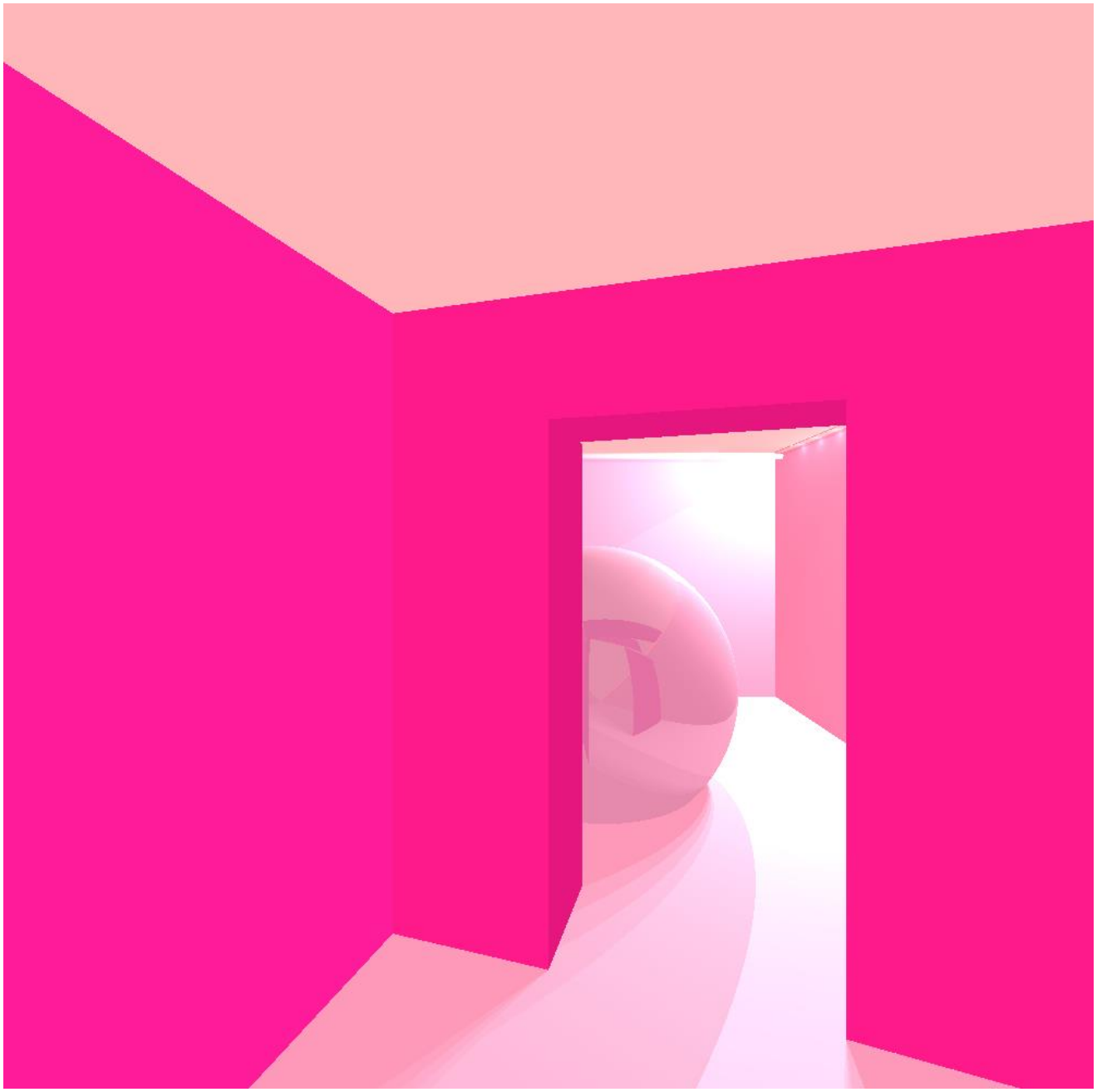
<u>הקוד של הלד</u>:
נגדיר את הלד להיות צילינדר עם שקיפות גבוהה כדי שהאור יוכל לעבור דרכו.
את האור עצמו נעשה עם לולאה שתעבור על הצילינדר ונוסיף pointLight במרחקים שווים .

```
double radius2 = 1;
Ray axisray2 = new Ray(new Point(32, 79, 0), new Vector( x: 0,  y: 0,  z: 1));
double length2 = 0.1;
Geometry ledRight = new Cylinder( radius2, axisray2, length2)
        .setEmission(new Color(RED))
        .setMaterial(new Material().setKd(0.5).setKs(0.5).setShininess(100).setKt(0.95));

//points lights on right wall
 for(int i = 0; i <40; i+=10) {
     scene1.lights.add(new PointLight(new Color(WHITE).scale( k: 0.6),
             new Point(34, 79, i),
              kC: 1,
              kL: 0.01111,
              kQ: 0.000111));
 }
```

בס"ד 27.06.23

התמונה הסופית



כמו שניתן לראות לתמונה שלנו יש צללים רבים כולם חדים בקצוות - נרצה לשפר זאת על ידי השיפור של צללים רכים soft shadows

עושים שליחה של כמה קרניים, אל נקודות סביב התאורה, כאשר אזור המטרה הוא עיגול ככה נראה הסטר-

```
ImageWriter imageWriter = new ImageWriter( imageName: "pinkRoomWithSoftShaddow",  nX: 1000,  nY: 1000);
camera3.setImageWriter(imageWriter)
        .setRayTracer(new RayTracerBasic(scene1).useSoftShadow( flag: true)
                .setNumOfSSRays(98).setRadiusBeamSS(10d) )
        .renderImage()
        .writeToImage();
```

נשלח 98 קרניים לכל פיקסל כאשר רדיוס אזור המטרה הוא 10 .

נוסיף למחלקה LightSource הצהרה על המתודה שתחשב את העיגול של אזור המטרה ,

```
/**
 * Creates a list of vectors from the given point to random points around the light within radius
 * for soft shadows
 *
 * @param p the given point
 * @param r the radius
 * @param amount the amount of vectors to create
 * @return list of vectors
 */
1 usage   2 implementations   Pazit1
List<Vector> getLCircle(Point p, double r, int amount);
```

נממש אותה במחלקה שלpointLight כי הרי הוא זה שיורש מ(LightSource)

```
//for random number of rays to create for soft shadows
2 usages
private static final Random RND = new Random();
1 usage   Pazit1 *
@Override
public List<Vector> getLCircle(Point p, double r, int amount) {
    if (p.equals(position))
        return null;

    List<Vector> result = new LinkedList<>();

    Vector l = getL(p); //vector to the center of the point light
    result.add(l);

    if (amount < 2) {
        return result;
    }

    Vector vAcross;
    //if l is parallel to z axis, then the normal is across z on x-axis
    if (isZero(l.getX()) && isZero(l.getY())) {
        //switch z and x places
        vAcross = new Vector( x: 0,  y: 0,  z: -1 * l.getZ()).normalize();
        //otherwise get the normal using x and y
    } else {//switched x and y places
        vAcross = new Vector(l.getX(), y: -1 * l.getY(),  z: 0).normalize();
    }
```

```
//the vector to the other direction
Vector vForward = vAcross.crossProduct(l).normalize();

double cosAngle, sinAngle, moveX, moveY, d;

for (int i = 0; i < amount; i++) {
    Point movedPoint = this.position;

    //random cosine of angle between (-1,1)
    cosAngle = 2 * RND.nextDouble() - 1;

    //sin(angle)=1-cos^2(angle)
    sinAngle = sqrt(1 - cosAngle * cosAngle);

    //d is between (-r,r)
    d = r * (2 * RND.nextDouble() - 1);
    //if we got 0 then try again, because it will just be the same as the center
    if (isZero(d)) {
        i--;
        continue;
    }
```

```
        //says how much to move across and down
        moveX = d * cosAngle;
        moveY = d * sinAngle;

        //moving the point according to the value
        if (!isZero(moveX)) {
            movedPoint = movedPoint.add(vAcross.scale(moveX));
        }
        if (!isZero(moveY)) {
            movedPoint = movedPoint.add(vForward.scale(moveY));
        }

        //adding the vector from the new point to the light position
        result.add(p.subtract(movedPoint).normalize());
    }
    return result;
```

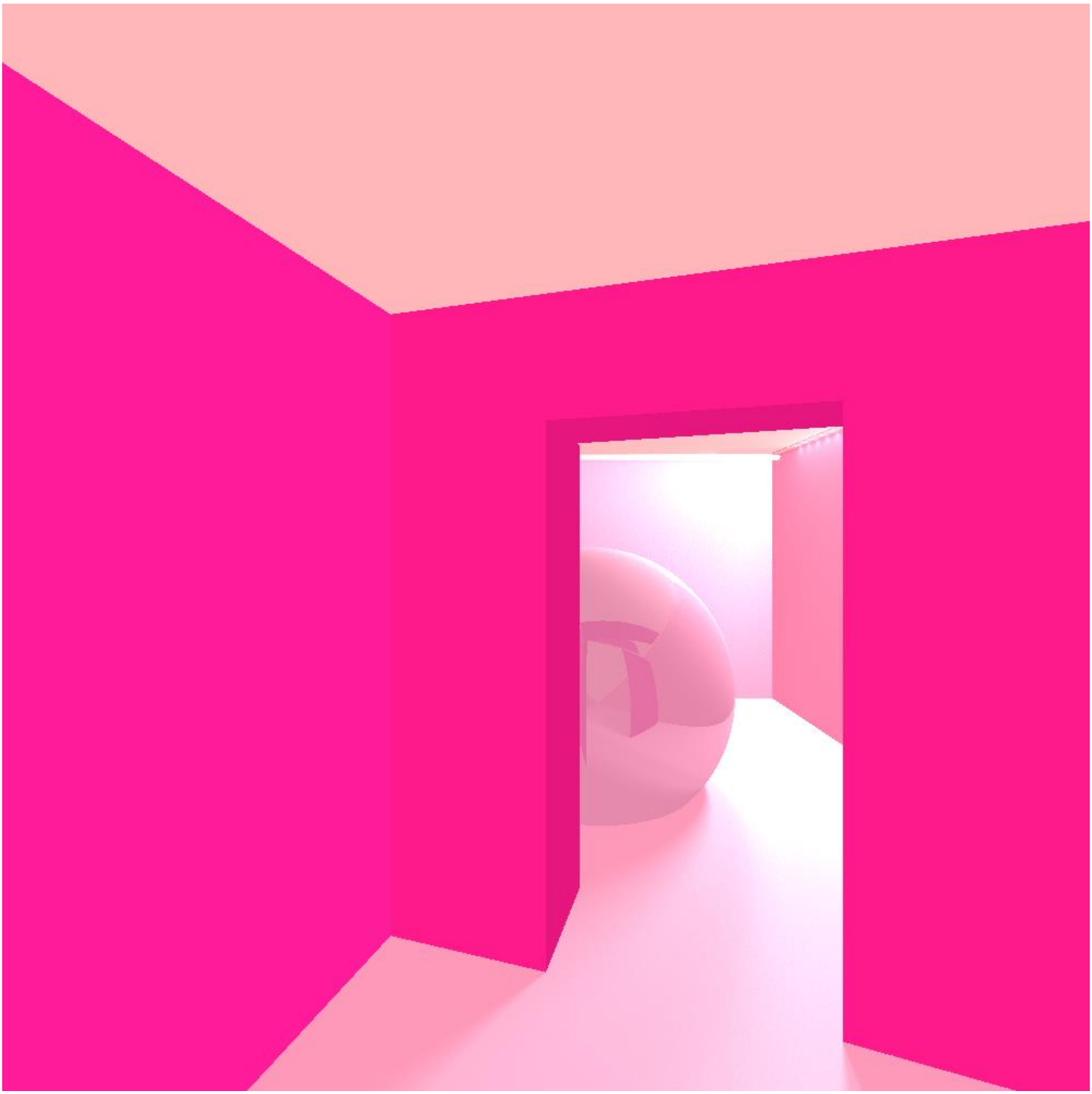
כעת במחלקה של RayTracerBasic נעדכן את הפונקציה של calcLocalEffects שתעשה בדיקה האם יש שימוש בשיפור או לא:

```
Material mat = geoPoint.getGeometry().getMaterial();
for (LightSource light : scene.lights) {
    Vector l = light.getL(geoPoint.getPoint());
    double nl = n.dotProduct(l);

    // Make sure that the perspective (camera) and the light source, are both in
    // the same side of the point's tangent plane sign(nl) == sign(nv)
    if (alignZero( number: nl * nv) > 0) {
        Double3 ktr = transparency(geoPoint, light, l, n);
        //==============================================
        //if soft shadow is not activated, get the regular transparency
        if (!isSoftShadow) {
            ktr = transparency(geoPoint, light, l, n);
            //otherwise get the transparency level according to soft shadow
        } else {
            ktr = transparencySS(geoPoint, light, n);
        }
        if (!(ktr.product(k)).lowerThan(MIN_CALC_COLOR_K)) {
            Color lightIntensity = light.getIntensity(geoPoint.getPoint()).scale(ktr);
            color = color.add(calcDiffusive(kD, l, n, lightIntensity),
                    calcSpecular(kS, l, n, v, nShininess, lightIntensity));
        }
    }
}
return color;
```

בס"ד 27.06.23

התמונה הסופית לאחר השיפור:



בס"ד 27.06.23

מבט מקרוב על השיפורים:

לפני:

אחרי :

לפני:

אחרי :