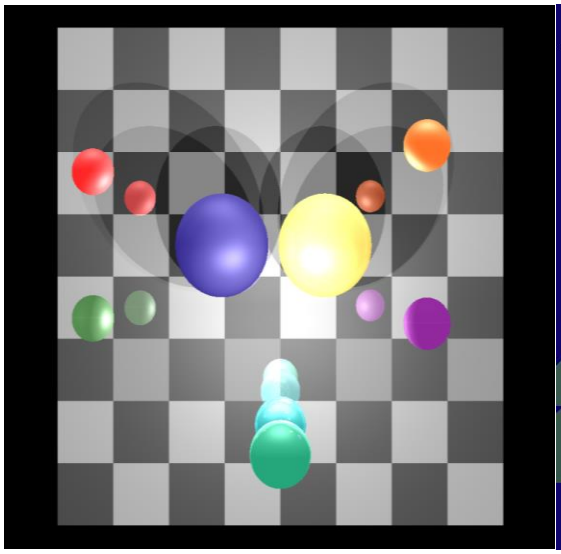
בס"ד

מיני פרויקט במבוא להנדסת תוכנה

151055.3.5781

Mini project in Introduction to Software Engineering

מגישות:
אנה זרביב- 340941244
קרן סממה-336351366



תודה גדולה למרצה אליעזר גינסבורגר!

בס"ד

תוכן עניינים

מיניפרויקט 1: שיפור תמונה

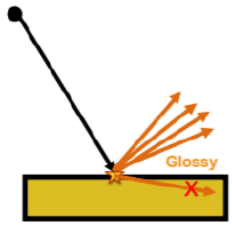
מיניפרויקט 2 : שיפור ביצועים

מיני פרויקט 1 : שיפור תמונה

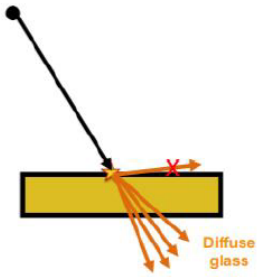
Glossy Surface and Diffuse Glass

1. הסבר הבעיה :

- Glossy surface
  - הבעיה - שזכוכית משקפת לחלוטין
  - הפתרון – במקום לשלוח קרן אחד של השתקפות נשלח יותר
  - תוצאה – מה שיותר קרוב יראה חד ומה שרחוק יותר יראה מטושטש בגלל שהתפזרות הקרניים מתרחבת עם המרחק



- Diffuse glass
  - הבעיה – משטחים שמבריקים באופן מושלם
  - הפתרון – במקום לשלוח קרן אחד של שקיפות נשלח יותר
  - תוצאה - מה שיותר קרוב יראה חד ומה שרחוק יותר יראה מטושטש בגלל שהתפזרות הקרניים מתרחבת עם המרחק



2. המימוש של השיפור

הוספנו פונקציה במחלקה Vector בשם ***normalToVector*** שמחשבת את הנורמל לווקטור. אצלנו מי שקוראת לפונקציה זאת תהיה הווקור של קרן השתקפות הראשון או קרן השקיפות הראשון שנוצר. בפונקציה, רוצים לאפס אחד הקואורדינטה ולהחליף סימן לאחרים אבל אם חלק הקואורדינטות אפסיות לא יכולות להיות כולם אפסיות לכן נאפס את הקואורדינטה הכי קטנה והסימן נחליף לאחד הגדולים.

```java
/**
 * This function helps us to calculate  a normal vector to the vector that calls the function
 * @return a new vector
 */
public Vector normalToVector()
{
    double coordinate;

    if(this.getHead().getX()>0) //finding the smallest coordinate of the vector to replace it with 0
    {
        coordinate = this.getHead().getX();
    }
    else {
        coordinate = -this.getHead().getX();
    }

    if(Math.abs(this.getHead().getY())<coordinate)
    {
        coordinate=1;
        if(this.getHead().getY()>0)
            coordinate=this.getHead().getY();
        else
            coordinate=-this.getHead().getY();
    }
    if(Math.abs(this.getHead().getZ())<coordinate)
    {
        coordinate=2;//last coordinate that we are checking so no need to reassign coordinate

    }
    if(coordinate==0) {//x is the smallest
        return new Vector( x: 0, -this.getHead().getZ(), this.getHead().getY()).normalize();
    }
    if(coordinate==1) {//y is the smallest
        return new Vector(-this.getHead().getZ(),  y: 0, this.getHead().getX()).normalize();
    }
    //z is the smallest
    return new Vector(this.getHead().getY(),-this.getHead().getX(), z: 0).normalize();
}
```

הוספנו במחלקה של קרן פונקציה בשם :
.createBeamOfRays

הפונקציה מייצרת אלומת קרניים לפי הקרן הראשית שהוא קרן שקיפות או קרן השתקפות. הפונקציה מקבלת כפרמטרים וקטור נורמל לראשית הקרן שקורא לפונקציה, מספר הקרניים שרוצים ליצור, והמרחק שאנחנו מחליטות ליצור מעגל וירטואלי שאנך לקרן שקורא לפונקציה. יצרנו שתי ווקטורים מאונכים לקרן הראשי : אחד בעזרת הפונקציה
normalToVector והשני בעזרת ה Cross-product ביניהם.
מפזרים נקודות רנדומליות בתוך המעגל ולפי זה מוצאים את הקרניים החדשות.
בסוף לקחנו בחשבון קרניים מעל המשטח בשביל קרני השתקפות ומתחת למשטח בשביל קרני שקיפות.

```java
/**
 * This function creates a beam of rays
 * @param normal of type Vector : normal vector to the head of the ray who calls the function
 * @param distance  of type double : the distance between the  point and the circle we are creating to find the beam
 * @param numOfRays of type int : the number of rays that will be in the beam
 * @return a list of all the rays int the beam
 */
public List<Ray>createBeamOfRays(Vector normal,double distance,int numOfRays)
{
    List<Ray> beam=new LinkedList<Ray>();
    beam.add(this);//the original ray that calls the function - there has to be at least one ray
    if(numOfRays==1) //if no additional rays were requested here
        return beam;

    Vector w=this.getDir().normalToVector();//finds a vector that is normal to the direction of the ray
    //the cross product between the normal vector w and the direction gives a new normal vector to the direction of the ray
    Vector v=this.getDir().crossProduct(w).normalize();

    Point3D center=this.getPoint(distance);//the center of our circle is the distance requested from p0, the head of the ray that calls the function
    Point3D randomP=Point3D.ZERO;
    double xRandom,yRandom,random;

    double randomRadiusValue = random(3 ,6); // the radius will be in range: 3 < r < 6, and will have double values


    for(int i=1;i<numOfRays;i++)//starts from 1 because there has to be at least one ray(the original)and we already add it to the beam
    {
        xRandom=random(-1,1);//random number [-1,1)
        yRandom=Math.sqrt(1-Math.pow(xRandom,2)); // toujours entre 0 et 1
        random=random(-randomRadiusValue,randomRadiusValue);
        if(xRandom!=0)//vector cannot be scaled with zero
            randomP=center.add(w.scale(xRandom*randomRadiusValue));//
        if(yRandom!=0)//vector cannot be scaled with zero
            randomP=center.add(v.scale(yRandom*randomRadiusValue));//randomRadiusValue

        Vector t= randomP.subtract(this.getP0());//vector from the head of the original ray and the random point

        double normalDotDir=alignZero(normal.dotProduct(this.getDir()));
        double normalDotT=alignZero(normal.dotProduct(t));
        if(normalDotDir*normalDotT>0)//if they are both positive or both negative then we need to create a ray with the original p0 and t
            beam.add(new Ray(this.getP0(),t));
    }
    return beam;
}
```

הוספנו במחלקת BasicRayTracer את המשתנים ***numOfRays_*** בשביל מספר הקרניים באלומת קרניים שרוצים ליצור ו***rayDistance_*** כדי לדעת מאיזה מרחק ליצור מעגל וירטואלי. שני הפרמטרים עם ערך ברירת מחדל אפס כדי שלא נצטרך לתקן את כל הטסטים הקודמים.

```java
/**
 * Parameters for ray tracing- glossy surface and diffuse glass - they are in class BasicRayTracer
 * because this class takes care of ray tracing.
 */
private int _numOfRays=0;
private double _rayDistance=0;
/**

/**
 * Set the distance between the intersection point and the circle
 * @param _rayDistance of type double
 */
public BasicRayTracer set_rayDistance(double _rayDistance) {
    if (_rayDistance < 0)
        throw new IllegalArgumentException("Distance cannot be negative");
    this._rayDistance = _rayDistance;
    return this;
}
/**
 * Get the distance we want between the intersection point and the circle
 * @return double for distance
 */

public double get_rayDistance() { return _rayDistance; }


/**
 * Set the number of rays that will be part of the beam
 * @param _numOfRays of type int :  amount of rays that will be part of the beam
 */
public BasicRayTracer set_numOfRays(int _numOfRays) {
    if (_numOfRays < 0)
        throw new IllegalArgumentException("Number of rays cannot be negative");
    this._numOfRays = _numOfRays;
    return this;
}
/**
 * Get number of rays of the beam
 * @return number of rays that will be part of the beam
 */
public int get_numOfRays() { return _numOfRays; }
```

כדי לתמוך בשיפורים, בסוף שינינו את הפונקציה
calcGlobalEffects
כי היא מחשבת את האפקים של קרני שקיפות והשתקפות.
הוספנו את היצירה של האלומת קרניים על שקיפות והשתקפות וביצענו קריאה רקורסיבית עבור כל קרן ששייך לאלומה והכנסנו את הצבע
למשתנה זמני ואחרי שסיימנו את הקריאות הרקורסיביות חילקנו במספר הקרניים ואז הוספנו לצבע.

```java
/**
 * This function takes account of global effects of interactions
 * between objects such as reflections and refractions of light.
 * This helps to be able to model transparent objects (with various
 * opacity levels) and reflecting surfaces such as mirrors.
 * @param geoPoint of type GeoPoint : the closest intersection point with the head of the ray from the camera
 * @param ray of type Ray : a ray from the camera
 * @param level the recursion level
 * @param k double - helps with recursion
 * @return color object from
 */
private Color calcGlobalEffects(GeoPoint geoPoint, Ray ray, int level, double k) {
    Color color = Color.BLACK;
    Color ReflectedColor = Color.BLACK;
    Color RefractedColor = Color.BLACK;
    Vector n = geoPoint._geometry.getNormal(geoPoint._point);
    Material material = geoPoint._geometry.getMaterial();
    List<Ray>beam1=new LinkedList<>(); // for beam of reflected ray
    List<Ray>beam2=new LinkedList<>(); // for beam of refracted ray

    double kkr = k * material.Kr;
    if (kkr > MIN_CALC_COLOR_K) { //if the reflection is bigger than the minimum of calc color

        Ray reflectionRay = constructReflectedRay(geoPoint,ray.getDir());
        //color = calcGlobalEffects(reflectionRay, level, material.Kr, kkr);
        if(this._numOfRays==0  || this._rayDistance<=0){
            beam1.add(reflectionRay);
        }
        else {
            beam1= reflectionRay.createBeamOfRays(geoPoint._geometry.getNormal(geoPoint._point), this.get_rayDistance(),this.get_numOfRays());
        }
        for(Ray r : beam1) // r = reflectedRay
        {
            ReflectedColor = ReflectedColor.add(calcGlobalEffects(r, level, material.Kr, kkr)); // //calls the recursion to find the rest of the color
        }
        if(beam1.size()>0) {
            color = color.add(ReflectedColor.reduce(beam1.size()));
        }
    }
```

```java
    double kkt = k * material.Kt;
    if (kkt > MIN_CALC_COLOR_K) { //if the refraction is bigger than the minimum of calc color
        Ray refractionRay = constructRefractedRay(geoPoint,ray.getDir());
        //color = color.add(calcGlobalEffects(refractionRay, level, material.Kt, kkt));
        if(this._numOfRays==0 ||this._rayDistance<=0)
            beam2.add(refractionRay);
        else{
            beam2= refractionRay.createBeamOfRays(geoPoint._geometry.getNormal(geoPoint._point), this.get_rayDistance(),this.get_numOfRays());
        }
        for(Ray r : beam2) // r = refractedRay
        {
            RefractedColor = RefractedColor.add(calcGlobalEffects(r, level, material.Kt,kkt)); // //calls the recursion to find the rest of the color
        }

        if(beam2.size()>0) {
            color = color.add(RefractedColor.reduce(beam2.size()));
        }
    }

    return color;

}
```

3. התוצאות :

הטסט שכתבנו כדי להראות את השיפורים. כל פעם שינינו מספר קרניים בעזרת
.Set_numOfRays()
ושיננו את השם של התמונה.

```java
public void finalProject1() {

    Scene scene2 = new Scene( name: "Test scene")//
            .setAmbientLight(new AmbientLight(new Color( r: 255,  g: 255,  b: 255),  Ka: 0.15)) //
            .setBackground(new Color( r: 0,  g: 0,  b: 0));


    scene2.geometries.add(new Sphere( radius: 20, new Point3D( x: 125,  y: -36,  z: 50))// violet
            .setEmission(new Color( r: 108,  g: 2,  b: 119))
            .setMaterial(new Material().setKd(0.3).setKs(0.2).setShininess(5).setKt(0).setKr(0)));

    scene2.geometries.add(new Sphere( radius: 20, new Point3D( x: 125,  y: 100,  z: 50))// orange
            .setEmission(new Color( r: 231,  g: 62,  b: 1))
            .setMaterial(new Material().setKd(0.25).setKs(0.3).setShininess(5).setKt(0.22).setKr(0)));

    scene2.geometries.add(new Sphere( radius: 25, new Point3D( x: 0,  y: -130,  z: -90))// bleu milieu en bas
            .setEmission(new Color( r: 0,  g: 128,  b: 128))
            .setMaterial(new Material().setKd(0.25).setKs(0.3).setShininess(5).setKt(0).setKr(0.3)));


    scene2.geometries.add(new Sphere( radius: 25, new Point3D( x: 0,  y: -130,  z: +90))//vert milieu dessus
            .setEmission(new Color( r: 0,  g: 128,  b: 85)) //0 128 85
            .setMaterial(new Material().setKd(0.25).setKs(0.3).setShininess(5).setKt(0).setKr(0.3)));

    scene2.geometries.add(new Sphere( radius: 50, new Point3D( x: -63,  y: +30,  z: -200))//BIG BLUE
            .setEmission(new Color( r: 15,  g: 5,  b: 107))
            .setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0)));


    scene2.geometries.add(new Sphere( radius: 50, new Point3D( x: 48,  y: 30,  z: -200))// BIG JAUNE
            .setEmission(new Color( r: 212,  g: 175,  b: 55))
            .setMaterial(new Material().setKd(0.3).setKs(0.2).setShininess(5).setKt(0).setKr(0.1)));

    scene2.geometries.add(new Sphere( radius: 20, new Point3D( x: -180,  y: 90,  z: -70))// red
            .setEmission(new Color( r: 255,  g: 0,  b: 0))
            .setMaterial(new Material().setKd(0.3).setKs(0.2).setShininess(5).setKr(0)));

    scene2.geometries.add(new Sphere( radius: 20, new Point3D( x: -180,  y: -36,  z: -70))// GREEN a gauche
            .setEmission(new Color( r: 34,  g: 80,  b: 30))
            .setMaterial(new Material().setKd(0.3).setKs(0.4).setShininess(5).setKt(0.22).setKr(0)));
```

```java
            .setMaterial(new Material().setKd(0.3).setKs(0.4).setShininess(5).setKt(0.22).setKr(0)));

//-----------------first line from the side right
scene2.geometries.add(new Polygon(new Point3D( x: 250,  y: 250,  z: -250),new Point3D( x: 187.5,  y: 250,  z: -250),
     new Point3D( x: 187.5,  y: 187.5,  z: -250),new Point3D( x: 250,  y: 187.5,  z: -250))
      .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 187.5,  y: 250,  z: -250),new Point3D( x: 125,  y: 250,  z: -250),
        new Point3D( x: 125,  y: 187.5,  z: -250),new Point3D( x: 187.5,  y: 187.5,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 125,  y: 250,  z: -250),new Point3D( x: 62.5,  y: 250,  z: -250),
        new Point3D( x: 62.5,  y: 187.5,  z: -250),new Point3D( x: 125,  y: 187.5,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 62.5,  y: 250,  z: -250),new Point3D( x: 0,  y: 250,  z: -250),
        new Point3D( x: 0,  y: 187.5,  z: -250),new Point3D( x: 62.5,  y: 187.5,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 0,  y: 250,  z: -250),new Point3D( x: -62.5,  y: 250,  z: -250),
        new Point3D( x: -62.5,  y: 187.5,  z: -250),new Point3D( x: 0,  y: 187.5,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -62.5,  y: 250,  z: -250),new Point3D( x: -125,  y: 250,  z: -250),
        new Point3D( x: -125,  y: 187.5,  z: -250),new Point3D( x: -62.5,  y: 187.5,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -125,  y: 250,  z: -250),new Point3D( x: -187.5,  y: 250,  z: -250),
        new Point3D( x: -187.5,  y: 187.5,  z: -250),new Point3D( x: -125,  y: 187.5,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -187.5,  y: 250,  z: -250),new Point3D( x: -250,  y: 250,  z: -250),
        new Point3D( x: -250,  y: 187.5,  z: -250),new Point3D( x: -187.5,  y: 187.5,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));
//-------------------------------second line
scene2.geometries.add(new Polygon(new Point3D( x: 250,  y: 187.5,  z: -250),new Point3D( x: 187.5,  y: 187.5,  z: -250),
        new Point3D( x: 187.5,  y: 125,  z: -250),new Point3D( x: 250,  y: 125,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));
scene2.geometries.add(new Polygon(new Point3D( x: 187.5,  y: 187.5,  z: -250),new Point3D( x: 125,  y: 187.5,  z: -250),
        new Point3D( x: 125,  y: 125,  z: -250),new Point3D( x: 187.5,  y: 125,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));
```

```java
scene2.geometries.add(new Polygon(new Point3D( x: 125,  y: 187.5,  z: -250),new Point3D( x: 62.5,  y: 187.5,  z: -250),
        new Point3D( x: 62.5,  y: 125,  z: -250),new Point3D( x: 125,  y: 125,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 62.5,  y: 187.5,  z: -250),new Point3D( x: 0,  y: 187.5,  z: -250),
        new Point3D( x: 0,  y: 125,  z: -250),new Point3D( x: 62.5,  y: 125,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 0,  y: 187.5,  z: -250),new Point3D( x: -62.5,  y: 187.5,  z: -250),
        new Point3D( x: -62.5,  y: 125,  z: -250),new Point3D( x: 0,  y: 125,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));
scene2.geometries.add(new Polygon(new Point3D( x: -62.5,  y: 187.5,  z: -250),new Point3D( x: -125,  y: 187.5,  z: -250),
        new Point3D( x: -125,  y: 125,  z: -250),new Point3D( x: -62.5,  y: 125,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -125,  y: 187.5,  z: -250),new Point3D( x: -187.5,  y: 187.5,  z: -250),
        new Point3D( x: -187.5,  y: 125,  z: -250),new Point3D( x: -125,  y: 125,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));
scene2.geometries.add(new Polygon(new Point3D( x: -187.5,  y: 187.5,  z: -250),new Point3D( x: -250,  y: 187.5,  z: -250),
        new Point3D( x: -250,  y: 125,  z: -250),new Point3D( x: -187.5,  y: 125,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

//-----------------------line 3

scene2.geometries.add(new Polygon(new Point3D( x: 250,  y: 125,  z: -250),new Point3D( x: 187.5,  y: 125,  z: -250),
        new Point3D( x: 187.5,  y: 62.5,  z: -250),new Point3D( x: 250,  y: 62.5,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 187.5,  y: 125,  z: -250),new Point3D( x: 125,  y: 125,  z: -250),
        new Point3D( x: 125,  y: 62.5,  z: -250),new Point3D( x: 187.5,  y: 62.5,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))


scene2.geometries.add(new Polygon(new Point3D( x: 125,  y: 125,  z: -250),new Point3D( x: 62.5,  y: 125,  z: -250),
        new Point3D( x: 62.5,  y: 62.5,  z: -250),new Point3D( x: 125,  y: 62.5,  z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))

scene2.geometries.add(new Polygon(new Point3D( x: 62.5,  y: 125,  z: -250),new Point3D( x: 0,  y: 125,  z: -250),
        new Point3D( x: 0,  y: 62.5,  z: -250),new Point3D( x: 62.5,  y: 62.5,  z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))
```

בס"ד

```
final project 1 ray.png × RenderTests.java × Main.java × final project 10 rays.png × Ray.java × BasicRayTracer.java × final project 5 rays.png × finalProject
            .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: 0, y: 125, z: -250),new Point3D( x: -62.5, y: 125, z: -250),
                new Point3D( x: -62.5, y: 62.5, z: -250),new Point3D( x: 0, y: 62.5, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: -62.5, y: 125, z: -250),new Point3D( x: -125, y: 125, z: -250),
                new Point3D( x: -125, y: 62.5, z: -250),new Point3D( x: -62.5, y: 62.5, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: -125, y: 125, z: -250),new Point3D( x: -187.5, y: 125, z: -250),
                new Point3D( x: -187.5, y: 62.5, z: -250),new Point3D( x: -125, y: 62.5, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: -187.5, y: 125, z: -250),new Point3D( x: -250, y: 125, z: -250),
                new Point3D( x: -250, y: 62.5, z: -250),new Point3D( x: -187.5, y: 62.5, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        //-----------------------------------line 4
    scene2.geometries.add(new Polygon(new Point3D( x: 250, y: 62.5, z: -250),new Point3D( x: 187.5, y: 62.5, z: -250),
                new Point3D( x: 187.5, y: 0, z: -250),new Point3D( x: 250, y: 0, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));


        scene2.geometries.add(new Polygon(new Point3D( x: 187.5, y: 62.5, z: -250),new Point3D( x: 125, y: 62.5, z: -250),
                new Point3D( x: 125, y: 0, z: -250),new Point3D( x: 187.5, y: 0, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: 125, y: 62.5, z: -250),new Point3D( x: 62.5, y: 62.5, z: -250),
                new Point3D( x: 62.5, y: 0, z: -250),new Point3D( x: 125, y: 0, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))


        scene2.geometries.add(new Polygon(new Point3D( x: 62.5, y: 62.5, z: -250),new Point3D( x: 0, y: 62.5, z: -250),
                new Point3D( x: 0, y: 0, z: -250),new Point3D( x: 62.5, y: 0, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: 0, y: 62.5, z: -250),new Point3D( x: -62.5, y: 62.5, z: -250),
                new Point3D( x: -62.5, y: 0, z: -250),new Point3D( x: 0, y: 0, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));
```

```
project 1 ray.png × RenderTests.java × Main.java × final project 10 rays.png × Ray.java × BasicRayTracer.java × final project 5 rays.png × finalProject.png × Vec
            .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: -62.5, y: 62.5, z: -250),new Point3D( x: -125, y: 62.5, z: -250),
                new Point3D( x: -125, y: 0, z: -250),new Point3D( x: -62.5, y: 0, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));


        scene2.geometries.add(new Polygon(new Point3D( x: -125, y: 62.5, z: -250),new Point3D( x: -187.5, y: 62.5, z: -250),
                new Point3D( x: -187.5, y: 0, z: -250),new Point3D( x: -125, y: 0, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: -187.5, y: 62.5, z: -250),new Point3D( x: -250, y: 62.5, z: -250),
                new Point3D( x: -250, y: 0, z: -250),new Point3D( x: -187.5, y: 0, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        //-----------------------------------line 5

        scene2.geometries.add(new Polygon(new Point3D( x: 250, y: 0, z: -250),new Point3D( x: 187.5, y: 0, z: -250),
                new Point3D( x: 187.5, y: -62.5, z: -250),new Point3D( x: 250, y: -62.5, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: 187.5, y: 0, z: -250),new Point3D( x: 125, y: 0, z: -250),
                new Point3D( x: 125, y: -62.5, z: -250),new Point3D( x: 187.5, y: -62.5, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: 125, y: 0, z: -250),new Point3D( x: 62.5, y: 0, z: -250),
                new Point3D( x: 62.5, y: -62.5, z: -250),new Point3D( x: 125, y: -62.5, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: 62.5, y: 0, z: -250),new Point3D( x: 0, y: 0, z: -250),
                new Point3D( x: 0, y: -62.5, z: -250),new Point3D( x: 62.5, y: -62.5, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: 0, y: 0, z: -250),new Point3D( x: -62.5, y: 0, z: -250),
                new Point3D( x: -62.5, y: -62.5, z: -250),new Point3D( x: 0, y: -62.5, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: -62.5, y: 0, z: -250),new Point3D( x: -125, y: 0, z: -250),
                new Point3D( x: -125, y: -62.5, z: -250),new Point3D( x: -62.5, y: -62.5, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));
```

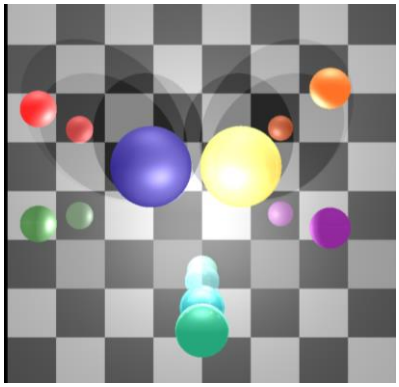
```
        scene2.geometries.add(new Polygon(new Point3D( x: -62.5, y: 0, z: -250),new Point3D( x: -125, y: 0, z: -250),
                new Point3D( x: -125, y: -62.5, z: -250),new Point3D( x: -62.5, y: -62.5, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: -125, y: 0, z: -250),new Point3D( x: -187.5, y: 0, z: -250),
                new Point3D( x: -187.5, y: -62.5, z: -250),new Point3D( x: -125, y: -62.5, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: -187.5, y: 0, z: -250),new Point3D( x: -250, y: 0, z: -250),
                new Point3D( x: -250, y: -62.5, z: -250),new Point3D( x: -187.5, y: -62.5, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

    //-----------------------------------line 6

        scene2.geometries.add(new Polygon(new Point3D( x: 250, y: -62.5, z: -250),new Point3D( x: 187.5, y: -62.5, z: -250),
                new Point3D( x: 187.5, y: -125, z: -250),new Point3D( x: 250, y: -125, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.1).setKs(0.1).setShininess(5).setKt(0).setKr(0.7)));

        scene2.geometries.add(new Polygon(new Point3D( x: 187.5, y: -62.5, z: -250),new Point3D( x: 125, y: -62.5, z: -250),
                new Point3D( x: 125, y: -125, z: -250),new Point3D( x: 187.5, y: -125, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

        scene2.geometries.add(new Polygon(new Point3D( x: 125, y: -62.5, z: -250),new Point3D( x: 62.5, y: -62.5, z: -250),
                new Point3D( x: 62.5, y: -125, z: -250),new Point3D( x: 125, y: -125, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.1).setKs(0.1).setShininess(5).setKt(0).setKr(0.7)))


        scene2.geometries.add(new Polygon(new Point3D( x: 62.5, y: -62.5, z: -250),new Point3D( x: 0, y: -62.5, z: -250),
                new Point3D( x: 0, y: -125, z: -250),new Point3D( x: 62.5, y: -125, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))


        scene2.geometries.add(new Polygon(new Point3D( x: 0, y: -62.5, z: -250),new Point3D( x: -62.5, y: -62.5, z: -250),
                new Point3D( x: -62.5, y: -125, z: -250),new Point3D( x: 0, y: -125, z: -250))
                .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.1).setKs(0.1).setShininess(5).setKt(0).setKr(0.7)))

        scene2.geometries.add(new Polygon(new Point3D( x: -62.5, y: -62.5, z: -250),new Point3D( x: -125, y: -62.5, z: -250),
                new Point3D( x: -125, y: -125, z: -250),new Point3D( x: -62.5, y: -125, z: -250))
                .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))
```

```java
.setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -125, y: -62.5, z: -250),new Point3D( x: -187.5, y: -62.5, z: -250),
        new Point3D( x: -187.5, y: -125, z: -250),new Point3D( x: -125, y: -125, z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.1).setKs(0.1).setShininess(5).setKt(0).setKr(0.7)));

scene2.geometries.add(new Polygon(new Point3D( x: -187.5, y: -62.5, z: -250),new Point3D( x: -250, y: -62.5, z: -250),
        new Point3D( x: -250, y: -125, z: -250),new Point3D( x: -187.5, y: -125, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));


//----------------------------------------line 7
scene2.geometries.add(new Polygon(new Point3D( x: 250, y: -125, z: -250),new Point3D( x: 187.5, y: -125, z: -250),
        new Point3D( x: 187.5, y: -187.5, z: -250),new Point3D( x: 250, y: -187.5, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 187.5, y: -125, z: -250),new Point3D( x: 125, y: -125, z: -250),
        new Point3D( x: 125, y: -187.5, z: -250),new Point3D( x: 187.5, y: -187.5, z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 125, y: -125, z: -250),new Point3D( x: 62.5, y: -125, z: -250),
        new Point3D( x: 62.5, y: -187.5, z: -250),new Point3D( x: 125, y: -187.5, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 62.5, y: -125, z: -250),new Point3D( x: 0, y: -125, z: -250),
        new Point3D( x: 0, y: -187.5, z: -250),new Point3D( x: 62.5, y: -187.5, z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 0, y: -125, z: -250),new Point3D( x: -62.5, y: -125, z: -250),
        new Point3D( x: -62.5, y: -187.5, z: -250),new Point3D( x: 0, y: -187.5, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -62.5, y: -125, z: -250),new Point3D( x: -125, y: -125, z: -250),
        new Point3D( x: -125, y: -187.5, z: -250),new Point3D( x: -62.5, y: -187.5, z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -125, y: -125, z: -250),new Point3D( x: -187.5, y: -125, z: -250),
        new Point3D( x: -187.5, y: -187.5, z: -250),new Point3D( x: -125, y: -187.5, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -187.5, y: -125, z: -250),new Point3D( x: -250, y: -125, z: -250),
```

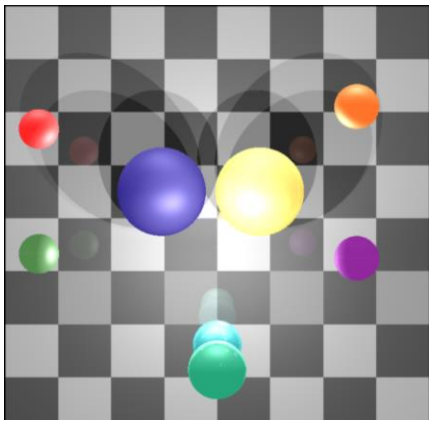
```java
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

//----------------------------------------line8

scene2.geometries.add(new Polygon(new Point3D( x: 250, y: -187.5, z: -250),new Point3D( x: 187.5, y: -187.5, z: -250),
        new Point3D( x: 187.5, y: -250, z: -250),new Point3D( x: 250, y: -250, z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 187.5, y: -187.5, z: -250),new Point3D( x: 125, y: -187.5, z: -250),
        new Point3D( x: 125, y: -250, z: -250),new Point3D( x: 187.5, y: -250, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 125, y: -187.5, z: -250),new Point3D( x: 62.5, y: -187.5, z: -250),
        new Point3D( x: 62.5, y: -250, z: -250),new Point3D( x: 125, y: -250, z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: 62.5, y: -187.5, z: -250),new Point3D( x: 0, y: -187.5, z: -250),
        new Point3D( x: 0, y: -250, z: -250),new Point3D( x: 62.5, y: -250, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))


scene2.geometries.add(new Polygon(new Point3D( x: 0, y: -187.5, z: -250),new Point3D( x: -62.5, y: -187.5, z: -250),
        new Point3D( x: -62.5, y: -250, z: -250),new Point3D( x: 0, y: -250, z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))


scene2.geometries.add(new Polygon(new Point3D( x: -62.5, y: -187.5, z: -250),new Point3D( x: -125, y: -187.5, z: -250),
        new Point3D( x: -125, y: -250, z: -250),new Point3D( x: -62.5, y: -250, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));

scene2.geometries.add(new Polygon(new Point3D( x: -125, y: -187.5, z: -250),new Point3D( x: -187.5, y: -187.5, z: -250),
        new Point3D( x: -187.5, y: -250, z: -250),new Point3D( x: -125, y: -250, z: -250))
        .setEmission(new Color( r: 96,  g: 96,  b: 96)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))


scene2.geometries.add(new Polygon(new Point3D( x: -187.5, y: -187.5, z: -250),new Point3D( x: -250, y: -187.5, z: -250),
        new Point3D( x: -250, y: -250, z: -250),new Point3D( x: -187.5, y: -250, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)))


//-----------------------------------------------------------------------------------------
```

```java
scene2.geometries.add(new Polygon(new Point3D( x: -187.5, y: -187.5, z: -250),new Point3D( x: -250, y: -187.5, z: -250),
        new Point3D( x: -250, y: -250, z: -250),new Point3D( x: -187.5, y: -250, z: -250))
        .setEmission(new Color( r: 0,  g: 0,  b: 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5)));


//-----------------------------------------------------------------------------------------
scene2.lights.add(new DirectionalLight(new Color( r: 240,  g: 240,  b: 240), new Vector( x: 0,  y: -1,  z: 0)));
scene2.lights.add(new PointLight(new Color( r: 170,  g: 170,  b: 170), new Point3D( x: -50,  y: -50,  z: -50)));
scene2.lights.add(new PointLight(new Color( r: 170,  g: 170,  b: 170), new Point3D( x: 50,  y: -50,  z: -50)));
scene2.lights.add(new SpotLight(new Color( r: 255, g: 255, b: 255), new Point3D( x: 0,  y: -30,  z: -100), new Vector( x: 0,  y: 1,  z: 0)) //
        .setKl(4E-5).setKq(2E-7));


ImageWriter imageWriter = new ImageWriter( imageName: "final project 50 rays",  nX: 1000,  nY: 1000);

Render render = new Render() //
        .setImageWriter(imageWriter) //
        .setCamera(camera2) //
        .setMultithreading(3)
        .setRayTracer(new BasicRayTracer(scene2).set_rayDistance(1).set_numOfRays(50));

render.renderImage();
render.writeToImage();

}
```

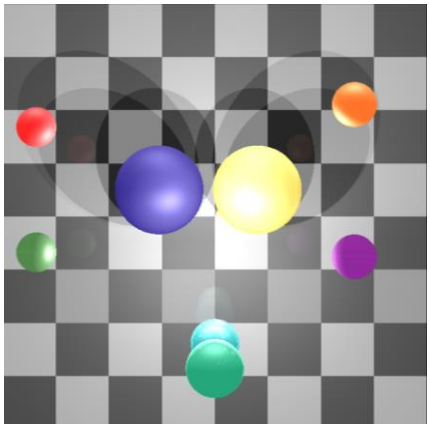
בלי שיפור קרן 1 : 14s 283 ms



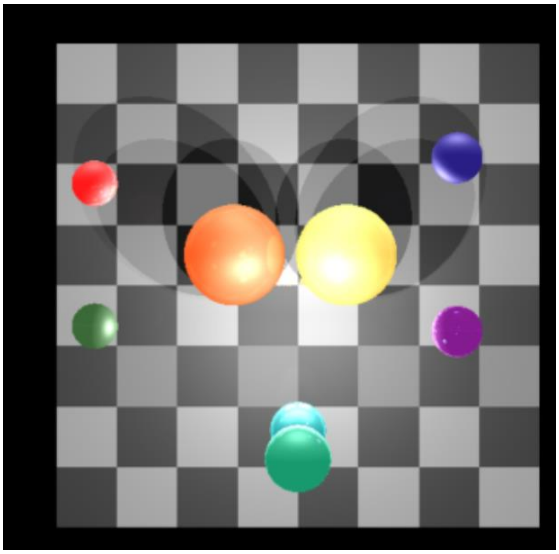
שיפור עם 5 קרניים : 23 s 201 ms



שיפור עם 10 קרניים : 59s 595 ms



שיפור עם 50 קרניים : 45 min 42s



מיני פרויקט 2 : שיפור ביצועים

Boundary Volume Hierarchy (BVH)

1. הספר הבעיה

מתחילת הפרוייקט , זרקנו המון קרניים וכל קרן בדקה חיתוך עם כל גאומטריות בסצנה , וזה בזבוז ממש רציני של זמן ריצה. לכן כדי לשפר אותו ניתן להשתמש בשיטת
עץ היררכי, BVH – boundary volume hierarchy . אנחנו
עוטפים מספר גאומטריות בקופסא, ובודקים האם הקרן פוגעת בקופסא . אם היא פוגעת, ניכנס ונבדוק אם היא חותכת את הגאומטריות שבפנים . אבל אם היא לא פוגעת בה , היא לא תיכנס בה בכלל.

2.מימוש השיפור

שינוי של הממשק
Intersectable
למחלקה אבסטרקטית שהוספנו בה מחלקה פנימית כדי ליצור קופסא מינימלית.
הקופסא מיוצגת על ידי נקודה מינימלית ונקודה מקסימלית. הוספנו גם קריאה לבנאי ברירת מחדל בשביל שכל צורה יקבל קופסה.
הוספנו שדה _setBoundingBox
עם פעולה של סטר כדי להפעיל שיפור לטסטים וערך ברירת מחדל שווה לשקר כדי לא לדרוס את הטסטים שלנו.

```java
Intersectable.java
public abstract class Intersectable{


    /**
     * _setBoundingBox is a parameter for BVH feature
     * It is true if we want to use a bounding box for improvement
     */
    public boolean _setBoundingBox = false;

    /**
     * Setter function for _setBoundingBox
     * @param setBoundingBox of type boolean
     */
    public  void setBoundingBox(boolean setBoundingBox) { _setBoundingBox = setBoundingBox; }

    /**
     * A static internal helper class to create a Box. A box is represented by a minimal point and a maximal point.
     */
    public static class Box {
        //makes them the opposite of what they should be so we can build boxes by checking if there is a bigger max or smaller min
        public double _maxX = Double.NEGATIVE_INFINITY;
        public double _minX = Double.POSITIVE_INFINITY;
        public double _maxY = Double.NEGATIVE_INFINITY;
        public double _minY = Double.POSITIVE_INFINITY;
        public double _maxZ = Double.NEGATIVE_INFINITY;
        public double _minZ = Double.POSITIVE_INFINITY;
    }

    // For each intersectable shape, it calls to the default constructor to build a box
    protected Box box = new Box();

```

פונקציה אבסטרקטית SetBoundingBox
שכל צורה יממש כדי ליצור קופסה מינימלית.

```java
    /**
     * a function that every intersectable geometry- should have in order to create
     * a bounding box
     */
    abstract public void setBoundingBox();
```

הוספנו פונקציה בוליאנית שבודקת אם קרן חותכת קופסה.

```java
/**
 *  This function checks if the ray intersects the box around a shape.
 * @param ray of type Ray
 * @return true if there is an intersection, otherwise return false.
 */
public boolean isIntersectionWithBox(Ray ray){

    Point3D start = ray.getP0();

    double start_X = start.getX();
    double start_Y = start.getY();
    double start_Z = start.getZ();

    Point3D direction = ray.getDir().getHead();

    double direction_X = direction.getX();
    double direction_Y = direction.getY();
    double direction_Z = direction.getZ();

    double max_t_for_X;
    double min_t_for_X;

    //If the direction_X is negative then the _min_X give the maximal value
    if (direction_X < 0) {
        max_t_for_X = (box._minX - start_X) / direction_X;
        // Check if the Intersectable is behind the camera
        if (max_t_for_X <= 0) return false;
        min_t_for_X = (box._maxX - start_X) / direction_X;
    }
    else if (direction_X > 0) {
        max_t_for_X = (box._maxX - start_X) / direction_X;
        if (max_t_for_X <= 0) return false;
        min_t_for_X = (box._minX - start_X) / direction_X;
    }
    else {
        if (start_X >= box._maxX || start_X <= box._minX)
            return false;
        else{
            max_t_for_X = Double.POSITIVE_INFINITY;
            min_t_for_X = Double.NEGATIVE_INFINITY;
        }
    }

    double max_t_for_Y;
    double min_t_for_Y;

    if (direction_Y < 0) {
        max_t_for_Y = (box._minY - start_Y) / direction_Y;
        if (max_t_for_Y <= 0) return false;
        min_t_for_Y = (box._maxY - start_Y) / direction_Y;
    }
    else if (direction_Y > 0) {
```

```
            max_t_for_Y = (box._maxY - start_Y) / direction_Y;
            if (max_t_for_Y <= 0) return false;
            min_t_for_Y = (box._minY - start_Y) / direction_Y;
        }
        else {
            if (start_Y >= box._maxY || start_Y <= box._minY)
                return false;
            else{
                max_t_for_Y = Double.POSITIVE_INFINITY;
                min_t_for_Y = Double.NEGATIVE_INFINITY;
            }
        }

        //Check the maximal and the minimal value for t
        double temp_max = Math.min(max_t_for_Y,max_t_for_X);
        double temp_min = Math.max(min_t_for_Y,min_t_for_X);
        temp_min = Math.max(temp_min,0);

        if (temp_max < temp_min) return false;

        double max_t_for_Z;
        double min_t_for_Z;

        if (direction_Z < 0) {
            max_t_for_Z = (box._minZ - start_Z) / direction_Z;
            if (max_t_for_Z <= 0) return false;
            min_t_for_Z = (box._maxZ - start_Z) / direction_Z;
        }
        else if (direction_Z > 0) {
            max_t_for_Z = (box._maxZ - start_Z) / direction_Z;
            if (max_t_for_Z <= 0) return false;
            min_t_for_Z = (box._minZ - start_Z) / direction_Z;
        }
        else {
            if (start_Z >= box._maxZ || start_Z <= box._minZ)
                return false;
            else{
                max_t_for_Z = Double.POSITIVE_INFINITY;
                min_t_for_Z = Double.NEGATIVE_INFINITY;
            }
        }

        temp_max = Math.min(max_t_for_Z,temp_max);
        temp_min = Math.max(min_t_for_Z,temp_min);

        if (temp_max < temp_min) return false;

        return true;
    }
```

עזרה מאתרים:

[Introduction to Acceleration Structures (Bounding Volume) (scratchapixel.com)](#)
[A Minimal Ray-Tracer: Rendering Simple Shapes (Sphere, Cube, Disk, Plane, etc.) (Ray-Box Intersection) (scratchapixel.com)](#)

במחלקה***Geometries***

שינינו את הבנאים ושיננו את פונקצית ההוספה של גיאומטריות כך שזה יחשב את הקופסאות על יותר מצורה אחת, על ידי קריאת לפונקציה SetBoundingBox

```java
/**
 * a function that every intersectable geometry- should have in order to create
 * a bounding box
 * Set the bounding boxes between shapes - if there is a bigger max value of smaller min value for one of the coordinates than what we already have we will switch it
 */
public void setBoundingBox()
{
    double xMinLimit = this.box._minX; // going in the negative direction of x
    double xMaxLimit = this.box._maxX; // going in the positive direction of x

    double yMinLimit = this.box._minY; // going in the negative direction of y
    double yMaxLimit = this.box._maxY; // going in the positive direction of y

    double zMinLimit = this.box._minZ; // going in the negative direction of z
    double zMaxLimit = this.box._maxZ; // going in the positive direction of z

    for(Object geometry : _intersectables)
    {
        if(geometry instanceof Geometry)
        {
            Geometry geo = ((Geometry)geometry);

            // setting x limits-------------------------
            if(geo.box._minX < xMinLimit)
                xMinLimit = geo.box._minX;

            if(xMaxLimit < geo.box._maxZ)
                xMaxLimit = geo.box._maxZ;

            // setting y limits-------------------------
            if(geo.box._minY< yMinLimit)
                yMinLimit = geo.box._minY;

            if(yMaxLimit < geo.box._maxY)
                yMaxLimit = geo.box._maxY;

            // setting z limits ----------------------
            if(geo.box._minZ < zMinLimit)
                zMinLimit = geo.box._minZ;

            if(zMaxLimit < geo.box._maxZ)
                zMaxLimit = geo.box._maxZ;
            //----------------------------------------
        }

        else
```

```java
ersectable.java   Geometries.java
                zMinLimit = geo.box._minZ;

                if(zMaxLimit < geo.box._maxZ)
                    zMaxLimit = geo.box._maxZ;
                //----------------------------------------
            }

            else
            {
                Geometries geo = ((Geometries)geometry); // in case there's an object of type Geometries- because of the hierarchy

                // setting x limits-------------------------
                if(geo.box._minX< xMinLimit)
                    xMinLimit = geo.box._minX;

                if(xMaxLimit < geo.box._maxZ)
                    xMaxLimit = geo.box._maxZ;

                // setting y limits-------------------------
                if(geo.box._minY< yMinLimit)
                    yMinLimit = geo.box._minY;

                if(yMaxLimit < geo.box._maxY)
                    yMaxLimit = geo.box._maxY;

                // setting z limits ----------------------
                if(geo.box._minZ < zMinLimit)
                    zMinLimit = geo.box._minZ;

                if(zMaxLimit < geo.box._maxZ)
                    zMaxLimit = geo.box._maxZ;
                //----------------------------------------

            }
        }

        this.box._maxX = xMaxLimit;
        this.box._minX = xMinLimit;
        this.box._maxY = yMaxLimit;
        this.box._minY = yMinLimit;
        this.box._maxZ = zMaxLimit;
        this.box._minZ =zMinLimit;

    }
```

התאמנו את הקוד בגאומטריות המתאימות שתהיה להם את האופציה לעבוד עם השיפור
במחלקת ***Sphere***

```java
/**
 * this function sets the values of the bounding box of the sphere
 */
@Override
public void setBoundingBox() {
    this.box._minX = _center.getX() - _radius; // x min
    this.box._maxX = _center.getX() + _radius; //x max
    this.box._minY = _center.getY() - _radius; // y min
    this.box._maxY = _center.getY() + _radius; // y max
    this.box._minZ = _center.getZ() - _radius; // z min
    this.box._maxZ = _center.getZ() + _radius; // z max
}
```

```java
/**
 * This function helps to find the intersections between a ray and a sphere.
 *
 * @param ray of type Ray
 * @return A list of intersection points of type GeoPoint or null if there are no intersections
 */
@Override
public List<GeoPoint> findGeoIntersections(Ray ray, double maxDistance) {

    if(_setBoundingBox==true && !isIntersectionWithBox(ray))// if the ray does not intersect the bounding box
    {
        return  null; // we will not calculate the intersection points with the sphere
    }

    Point3D p0 = ray.getP0(); // beginning point of the ray
    Vector v = ray.getDir();  // vector direction of the ray
    Point3D O = _center;      // center of the sphere

    Vector u;
    try {
```

במחלקת ***Polygon***

```java
/**
 * this function sets the values of the bounding box of the polygon
 */
@Override
public void setBoundingBox() {
    //starting points
    box._minX = this._vertices.get(0).getX();
    box._maxX = this._vertices.get(0).getX();
    box._minY = this._vertices.get(0).getY();
    box._maxY = this._vertices.get(0).getY();
    box._minZ = this._vertices.get(0).getZ();
    box._maxZ = this._vertices.get(0).getZ();

    for (int j = 1; j < this._vertices.size(); j++) {
        if (this._vertices.get(j).getX() < box._minX)//checks for min x
            box._minX = this._vertices.get(j).getX();
        if (this._vertices.get(j).getY() < box._minY)//checks for min y
            box._minY = this._vertices.get(j).getY();
        if (this._vertices.get(j).getZ() < box._minZ)//checks for min z
            box._minZ = this._vertices.get(j).getZ();
        if (this._vertices.get(j).getX() > box._maxX)//checks for max x
            box._maxX = this._vertices.get(j).getX();
        if (this._vertices.get(j).getY() > box._maxY)//checks for max y
            box._maxY = this._vertices.get(j).getY();
        if (this._vertices.get(j).getZ() > box._maxZ)//checks for max z
            box._maxZ = this._vertices.get(j).getZ();
    }
```

```java
/**
 * This function helps to find the intersection between a ray and a polygon.
 * Fist, we check intersection between the ray and the plane. If there is an intersection point,
 * we must update the geometry field from the GeoPoint originally received from the plane
 * to the value "this" related to the polygon
 *
 * @param ray of type Ray
 * @return a list of intersection points of type GeoPoint or null if there are no intersections
 */
@Override
public List<GeoPoint> findGeoIntersections(Ray ray, double maxDistance) {


    if(_setBoundingBox==true && !isIntersectionWithBox(ray))// if the ray does not intersect the bounding box
    {

        return  null; // we will not calculate the intersection points with the polygon
    }


    List<GeoPoint> intersections = _plane.findGeoIntersections(ray, maxDistance);
    if (intersections == null) {
        return null;
    }
```

3.תוצאות

התמונה שלנו עם שליחת :

-קרן 1 :10 s 302 ms

-5 קרניים : 16s 997 ms

-10 קרניים : 41s 371ms

סיכום

במהלך הפרוייקט, יצרנו גאומטריות, אלמנטים, פרימיטיביים, מצלמה, רנדר ועוד מחלקות רבות על מנת ליצור סצנות. השתדלנו לתת שמות משמעותיים למשתנים, הוספנו תיעוד להבנה. שיפרנו את התמונה בעזרת

Glossy surface and Diffuse glass.

זמן הריצה השתפר משמעותית עם הוספת שיפור של BVH.