

**מגישים:**

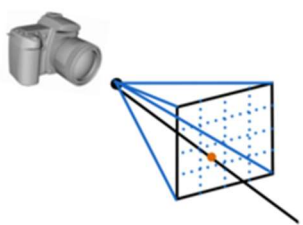
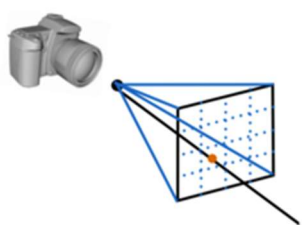
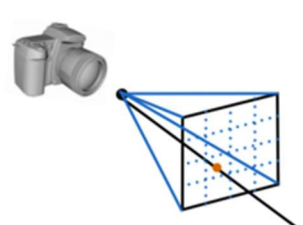
נועם שוובר (214120032)

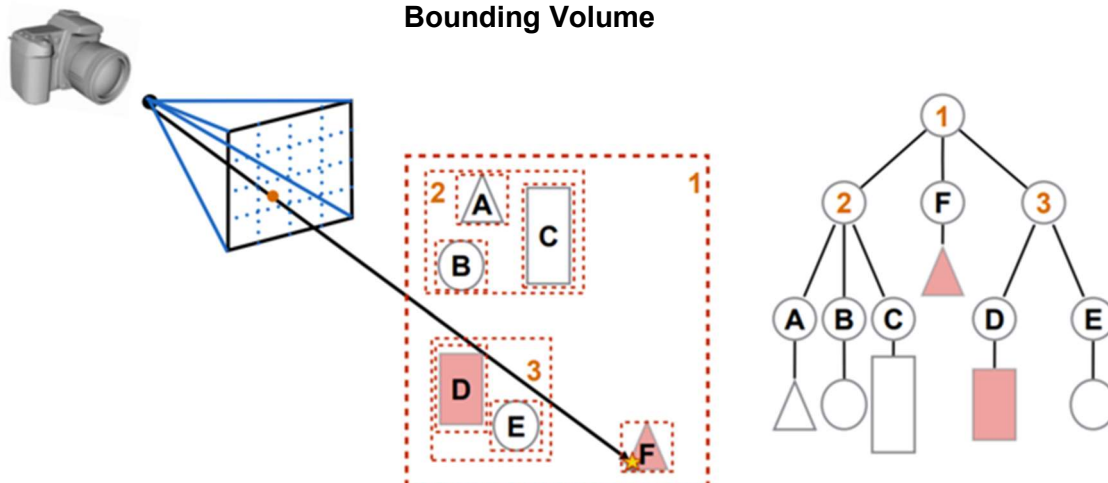
ישעיה צובל (325889160)

**דו"ח – מיני פרויקט 2**

- הוספנו ממשק Boundable (ניתן לחסימה) וגרמנו לכל הגיאומטריות הסופיות לרשת ממנו.
- יצרנו מחלקה AxisAlignedBoundingBox (קופסא חוסמת על הצירים) שמממשת גם את המחלקות Boundable ו-Intersectable.
- הוספנו הערות Javadoc והערות כלליות רבות על מנת לתעד כל חלק בקוד שיהיה מובן מה מטרתו ומה פעולתו.
- שיפורי הביצועים שלנו:
- MultiThreading – ריבוי תהליכים.
- Bouding Volume Heirarchy – היררכית קופסאות חוסמות.

**MultiThreading**

Thread 1	Thread 2	Thread 3
		

**Bounding Volume**

**מגישים:**

נועם שוובר (214120032)

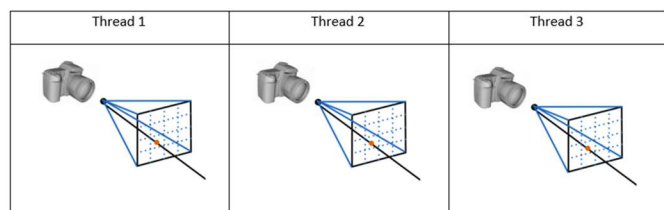
ישעיה צובל (325889160)

הסבר על השיפורים*Multi-Threading*

✓	✓ Test Results	1 min 42 sec
✓	✓ MinipOneTests	1 min 42 sec
	✓ AATest()	1 min 42 sec

הבעיה בפרויקט שלנו עד כה היא שבגלל כל השיפורים ממני פרויקט 1 שמוסיפים עוד הרבה חישובים של חיתוכי קרניים, זמן הריצה של הרינדור הוא מאוד ארוך. (ניתן לראות, זמן ריצה של הטסט של AntiAliasing שלנו עם 5\*5 קרניים לוקח 1 דקה ו 42 שניות).

MultiThreading



על מנת לתקן זאת, מחלקים את פעולת הרינדור בין כמה תהליכונים שונים, כך שבמקום שכל חיתוכי הקרניים יחושבו אחד אחרי השני, נחשב אותם במקביל בחלוקת עבודה בין כמה תהליכונים.

**מגישים:**

נועם שוובר (214120032)

ישעיה צובל (325889160)

**הקוד שלנו**

במחלקה Render פונקציה לקביעה בכמה תהליכונים נשתמש:

```

/**
 * Set multi-threading <br>
private class Pixel {
    private long maxRows = 0;
    private long maxCols = 0;
    private long pixels = 0;
    public volatile int row = 0;
    public volatile int col = -1;
    private long counter = 0;
    private int percents = 0;
    private long nextCounter = 0;

    /**
     * The constructor for initializing the main follow up Pixel object
     *
     * @param maxRows the amount of pixel rows
     * @param maxCols the amount of pixel columns
     */
    public Pixel(int maxRows, int maxCols) {
        this.maxRows = maxRows;
        this.maxCols = maxCols;
        this.pixels = (long) maxRows * maxCols;
        this.nextCounter = this.pixels / 100;
        if (Render.this.print)
            System.out.printf("\r %02d%%", this.percents);
    }

```

במחלקה Render תת מחלקה Pixel להגדרת קטע עבודה של הרינדור שצריך להיעשות:

## מגישים:

נועם שוובר (214120032)

ישעיה צובל (325889160)

במחלקה Render תת המחלקה Pixel, פונקציה לבחירה והחזרה של קטע הרינדור הבא לחישוב (הפיקסל הבא):

```
private synchronized int nextP(Pixel target) {
    ++col;
    ++this.counter;
    if (col < this.maxCols) {
        target.row = this.row;
        target.col = this.col;
        if (Render.this.print && this.counter == this.nextCounter) {
            ++this.percents;
            this.nextCounter = this.pixels * (this.percents + 1) / 100;
            return this.percents; }
        return 0;
    }
    ++row;
    if (row < this.maxRows) {
        col = 0;
        target.row = this.row;
        target.col = this.col;
        if (Render.this.print && this.counter == this.nextCounter) {
            ++this.percents;
            this.nextCounter = this.pixels * (this.percents + 1) / 100;
            return this.percents; }
        return 0;
    }
    return -1;
}

public boolean nextPixel(Pixel target) {
    int percent = nextP(target);
    if (Render.this.print && percent > 0)
        synchronized (this) {
            notifyAll();
        }
    if (percent >= 0)
        return true;
    if (Render.this.print)
        synchronized (this) {
            notifyAll();
        }
    return false;
}
```

**מגישים:**

נועם שוובר (214120032)

ישעיה צובל (325889160)

במחלקה Render, פונקציה לרינדור בעזרת התהליכונים:

```

private void renderImageThreaded() {
    final int nX = imageWriter.getNx();
    final int nY = imageWriter.getNy();
    final Pixel thePixel = new Pixel(nY, nX);
    // Generate threads
    Thread[] threads = new Thread[threadsCount];
    for (int i = threadsCount - 1; i >= 0; --i) {
        threads[i] = new Thread(() -> {
            Pixel pixel = new Pixel();
            while (thePixel.nextPixel(pixel))
                castRays(nX, nY, pixel.col, pixel.row); });
    }
    // Start threads
    for (Thread thread : threads)
        thread.start();

    // Print percents on the console
    thePixel.print();

    // Ensure all threads have finished
    for (Thread thread : threads)
        try {
            thread.join();
        } catch (Exception e) { }

    if (print)
        System.out.print("\r100%");
}

```

**מגישים:**

נועם שוובר (214120032)

ישעיה צובל (325889160)

**דוגמא לשיפור**

כפי שראינו, הרצת דוגמא מיני פרויקט 1 שלנו עם AntiAliasing של 5 קרניים ותהליכון 1, יוצא דקה ו 42 שניות זמן ריצה.

✓	Test Results	1 min 42 sec
✓	MinipOneTests	1 min 42 sec
✓	AATest()	1 min 42 sec

בהרצה עם 3 תהליכונים, זמן הריצה החדש הוא 59 שניות – שיפור של כמעט פי 2.

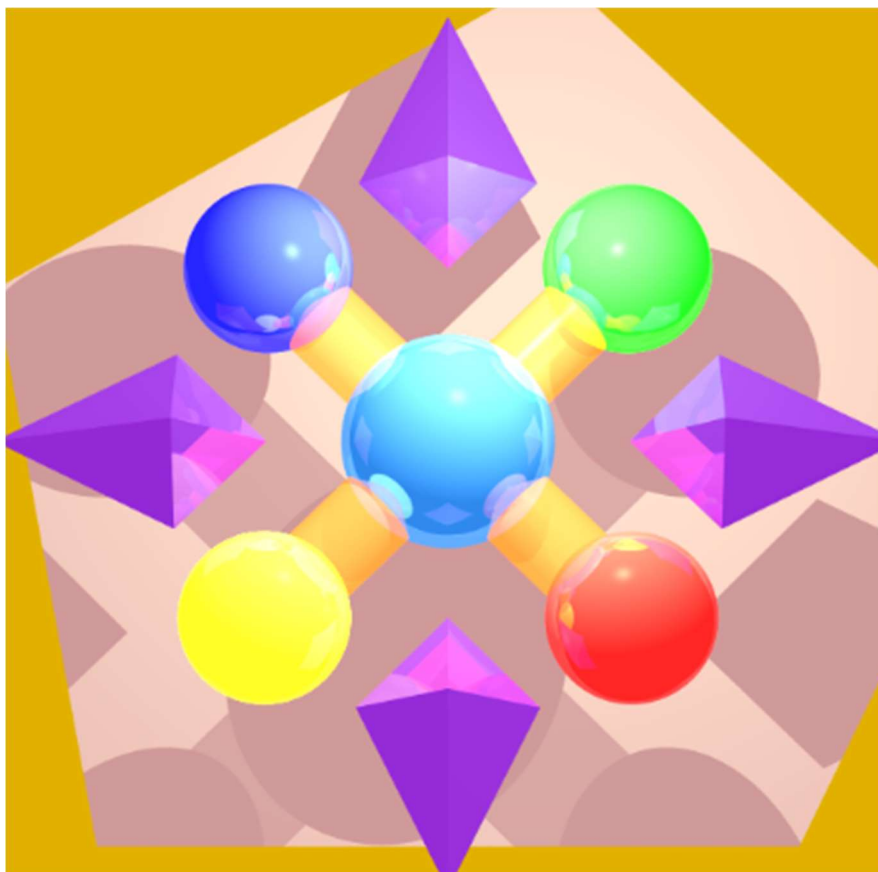
✓	Test Results	59 sec 60 ms
✓	MinipOneTests	59 sec 60 ms
✓	AATest()	59 sec 60 ms

**מגישים:**

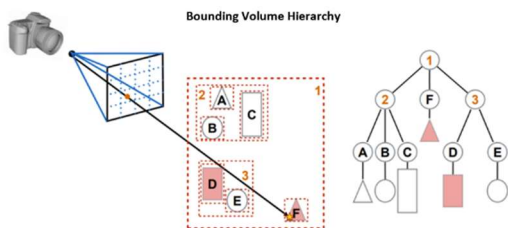
נועם שוובר (214120032)

ישעיה צובל (325889160)

התמונה שיצאה:

*Bounding Volume Hierachy*

למרות שחסכנו את חלק מזמן הריצה על ידי יצירת תהליכונים וחלוקת העבודה ביניהם, עדיין זמן הריצה עבור סצנות מרובות גיאומטריות הוא ארוך מאוד. (ניתן לראות שלהריץ את הסצנה Teapot עם 3 תהליכונים ו AntiAliasing של 5 קרניים לוקח עדיין הרבה מאוד זמן).



על מנת לשפר אפילו יותר את זמני הריצה, בונים היררכיה של קופסאות חוסמות של הגיאומטריות בסצנה. כל קרן, תבדוק בכל פעם האם היא נחתכת בקופסא החוסמת הכללית וכך תרד בהיררכיה ותבדוק האם היא נחתכת רק עבור גיאומטריות קרובות שיש סיכוי שנחתך בהם. כך, נחסוך הרבה בדיקות מיותרות של חיתוכי קרניים עם סצנות שבטוח לא יגיעו אליהם.

**מגישים:**

נועם שוובר (214120032)

ישעיה צובל (325889160)

**הקוד שלנו**

הוספנו ממשק חדש Boundable, שמטרתו לבצע פעולות של חסימת גיאומטריות:

```
/**
 * An interface to take care of creating Bounding Boxes
 */
public interface Boundable {

    /**
     * Creates a box around the object, adds the object to its list.
     * @return The bounding box of the object
     */
    AxisAlignedBoundingBox getBoundingBox();
}
```

בכל גיאומטריה סופית, הוספנו מימוש של הפונקציה שמחזירה קופסא חוסמת בממשק לדוגמא בכדור:

```
@Override
public AxisAlignedBoundingBox getBoundingBox() {
    AxisAlignedBoundingBox res = new AxisAlignedBoundingBox( minX: getCenter().getCx()-radius,
        minY: getCenter().getCy()-radius,
        minZ: getCenter().getCz()-radius,
        maxX: getCenter().getCx()+radius,
        maxY: getCenter().getCy()+radius,
        maxZ: getCenter().getCz()+radius);
    res.addToContains( boundable: this);

    return res;
}
```



**מגישים:**

נועם שוובר (214120032)

ישעיה צובל (325889160)

יצרנו מחלקה למימוש קופסאות חוסמות:

```

public class AxisAlignedBoundingBox implements Intersectable, Boundable {
    /**
     * The minimum values of the box on the axis
     */
    private double minX, minY, minZ;

    /**
     * The maximum values of the box on the axis
     */
    private double maxX, maxY, maxZ;

    /**
     * The middle of the box on the axis
     */
    private double midX, midY, midZ;

    /**
     * A list of the contained boundable objects
     */
    private List<Boundable> contains;

```

בנאי של קופסא על ידי הצבת ערכי הקצה של הקופסא:

```

public AxisAlignedBoundingBox(double minX, double minY, double minZ, double maxX, double maxY, double maxZ) {
    this.minX = minX;
    this.minY = minY;
    this.minZ = minZ;

    this.maxX = maxX;
    this.maxY = maxY;
    this.maxZ = maxZ;

    this.midX = (minX + maxX) / 2;
    this.midY = (minY + maxY) / 2;
    this.midZ = (minZ + maxZ) / 2;

    contains = new ArrayList<>();
}

```

**מגישים:**

נועם שוובר (214120032)

ישעיה צובל (325889160)

בנאי הבונה קופסא החוסמת רשימה של קופסאות:

```
public AxisAlignedBoundingBox(List<AxisAlignedBoundingBox> boxes) {
    this.maxX = boxes.get(0).getMaxX();
    this.maxY = boxes.get(0).getMaxY();
    this.maxZ = boxes.get(0).getMaxZ();
    this.minX = boxes.get(0).getMinX();
    this.minY = boxes.get(0).getMinY();
    this.minZ = boxes.get(0).getMinZ();

    for (int i = 1; i < boxes.size(); i++) {
        if (boxes.get(i).getMaxX() > maxX) {
            this.maxX = boxes.get(i).getMaxX();
        }
        if (boxes.get(i).getMaxY() > maxY) {
            this.maxY = boxes.get(i).getMaxY();
        }
        if (boxes.get(i).getMaxZ() > maxZ) {
            this.maxZ = boxes.get(i).getMaxZ();
        }
        if (boxes.get(i).getMinX() < minX) {
            this.minX = boxes.get(i).getMinX();
        }
        if (boxes.get(i).getMinY() < minY) {
            this.minY = boxes.get(i).getMinY();
        }
        if (boxes.get(i).getMinZ() < minZ) {
            this.minZ = boxes.get(i).getMinZ();
        }
    }

    this.midX = (minX + maxX) / 2;
    this.midY = (minY + maxY) / 2;
    this.midZ = (minZ + maxZ) / 2;

    contains = new ArrayList<>();
}
```

## מגישים:

נועם שוובר (214120032)

ישעיה צובל (325889160)

פונקציה לבניית עץ היררכי של קופסאות על ידי קבלת רשימה של איברים ניתנים לחסימה:

```
public static AxisAlignedBoundingBox createTree(List<Boundable> boundables) {
    //if we got 0 boundables to bound
    if (boundables.size() == 0)
        return null;

    else {
        //turn the list of boundables into a list of boxes that encapsulate the
        boundables
        ArrayList<AxisAlignedBoundingBox> boxes = new
        ArrayList<>(boundables.stream().map(boundable ->
        boundable.getBoundingBox()).collect(Collectors.toList()));
        return createTreeRec(boxes);
    }
}

private static AxisAlignedBoundingBox
createTreeRec(ArrayList<AxisAlignedBoundingBox> boxes) {
    //create a box that encapsulates all the other ones
    AxisAlignedBoundingBox node = new AxisAlignedBoundingBox(boxes);

    //find the longest edge of the box
    double x = node.maxX - node.minX;
    double y = node.maxY - node.minY;
    double z = node.maxZ - node.minZ;
    int edge = x > y && x > z ? 0 : (y > x && y > z ? 1 : 2);

    //base of the recursion, if the list has 1 box, return it
    if (boxes.size() == 1) {
        return boxes.get(0);
    }

    //base of the recursion, if the list has 2 boxes
    if (boxes.size() <= 2) {
        for (AxisAlignedBoundingBox box : boxes) {
            node.addToContains(box); //add them to this box
        }
    }

    //recursion step, if we have more boxes left
    else {
        ArrayList<AxisAlignedBoundingBox> left, right;

        //sort the boxes according to how they align on that edge
        sortBoxesByAxis(boxes, edge);

        //split the list into 2 even pieces
        left = new ArrayList<>(boxes.subList(0, boxes.size() / 2));
        right = new ArrayList<>(boxes.subList(boxes.size() / 2, boxes.size()));

        //go down the recursion
        node.addToContains(createTreeRec(left));
        node.addToContains(createTreeRec(right));
    }

    return node;
}
```

## מגישים:

נועם שוובר (214120032)

ישעיה צובל (325889160)

פונקציה המחזירה חיתוכים של קרן עם הגיאומטריות החסומות בסצנה:

```

public List<GeoPoint> findGeoIntersections(Ray ray, double maxDistance) {
    //the ray's head and direction points
    Point3D dir = ray.getDir().getHead();
    Point3D point = ray.getPO();

    double xMax, yMax, zMax, xMin, yMin, zMin;

    //if the vector's x coordinate is zero
    if (isZero(dir.getCx())) {
        //if the point's x value is in the box,
        if (maxX >= point.getCx() && minX <= point.getCx()) {
            xMax = Double.MAX_VALUE;
            xMin = Double.MIN_VALUE;
        } else
            return null;
    }

    //if the vector's x coordinate is not zero, we need to check if we have values
    //where (MaxX - pointX) / dirX > (MinX - pointX) / dirX
    else {
        double t1 = (maxX - point.getCx()) / dir.getCx();
        double t2 = (minX - point.getCx()) / dir.getCx();
        xMin = Math.min(t1, t2);
        xMax = Math.max(t1, t2);
    }

    //if the vector's y coordinate is zero
    if (isZero(dir.getCy())) {
        //if the point's y value is in the box,
        if (maxY >= point.getCy() && minY <= point.getCy()) {
            yMax = Double.MAX_VALUE;
            yMin = Double.MIN_VALUE;
        } else
            return null;
    }

    //if the vector's y coordinate is not zero, we need to check if we have
    values
    //where (MaxY - pointY) / dirY > (MinY - pointY) / dirY
    else {
        double t1 = (maxY - point.getCy()) / dir.getCy();
        double t2 = (minY - point.getCy()) / dir.getCy();
        yMin = Math.min(t1, t2);
        yMax = Math.max(t1, t2);
    }

    //if the vector's z coordinate is zero
    if (isZero(dir.getCz())) {
        //if the point's z value is in the box,
        if (maxZ >= point.getCz() && minZ <= point.getCz()) {
            zMax = Double.MAX_VALUE;
            zMin = Double.MIN_VALUE;
        } else
            return null;
    }
}

```

## מגישים:

נועם שוובר (214120032)

ישעיה צובל (325889160)

```
//if the vector's z coordinate is not zero, we need to check if we have
values
//where (Max2 - point2) / dir2 > (Min2 - point2) / dir2
else {
    double t1 = (max2 - point.getCz()) / dir.getCz();
    double t2 = (min2 - point.getCz()) / dir.getCz();
    zMin = Math.min(t1, t2);
    zMax = Math.max(t1, t2);
}

//check if such a point exists.
if (xMin > yMax || xMin > zMax ||
    yMin > xMax || yMin > zMax ||
    zMin > yMax || zMin > xMax)
    return null; //if not return null

//if they do, return all the intersection points of the contents of
the box
else {
    List<GeoPoint> lst = new ArrayList<>();
    for (Boundable geo : contains) {
        List<GeoPoint> pointLst = ((Intersectable)
geo).findGeoIntersections(ray, maxDistance);
        if (pointLst != null)
            lst.addAll(pointLst);
    }

    return lst;
}
}
```

**מגישים:**

נועם שוובר (214120032)

ישעיה צובל (325889160)

**דוגמא לשיפור**

✓	✓ Test Results	35 min 38 sec
✓	✓ TeapotTest	35 min 38 sec
✓	✓ teapot1()	35 min 38 sec

כפי שראינו, הרצת הדוגמא של Teapot שלנו עם AntiAliasing של 5 קרניים ושלושה תהליכונים, יוצא 35 דקות ו 38 שניות זמן ריצה.

✓	✓ Test Results	2 min 38 sec
✓	✓ TeapotTest	2 min 38 sec
✓	✓ teapot1()	2 min 38 sec

בהרצה עם BVH, זמן הריצה החדש הוא 2 דקות ו 38 שניות – שיפור ענק של כ-פי 13.5.

**התמונה שיצאה:**