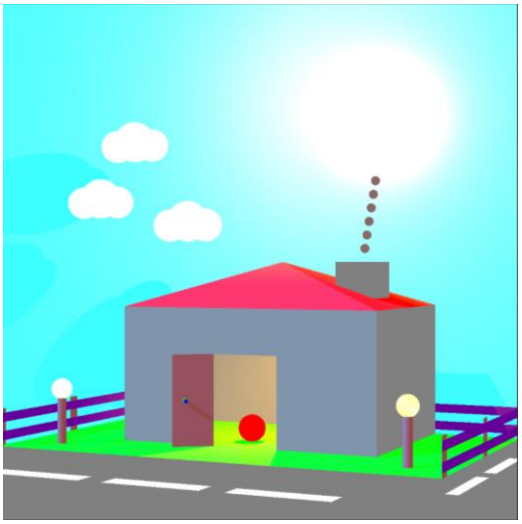
בס"ד

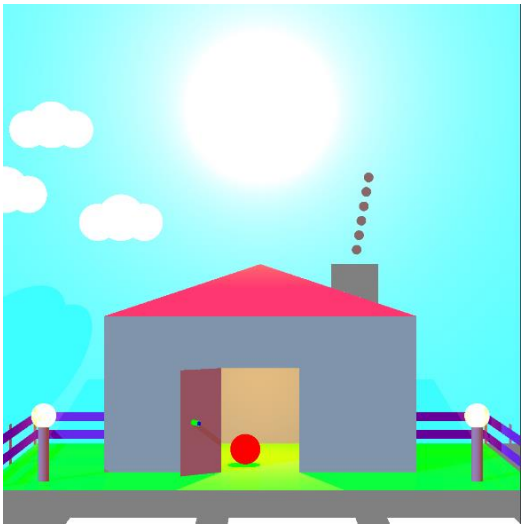
דניאל שלום כהן – 212991749

נתן שטרן – 322879255

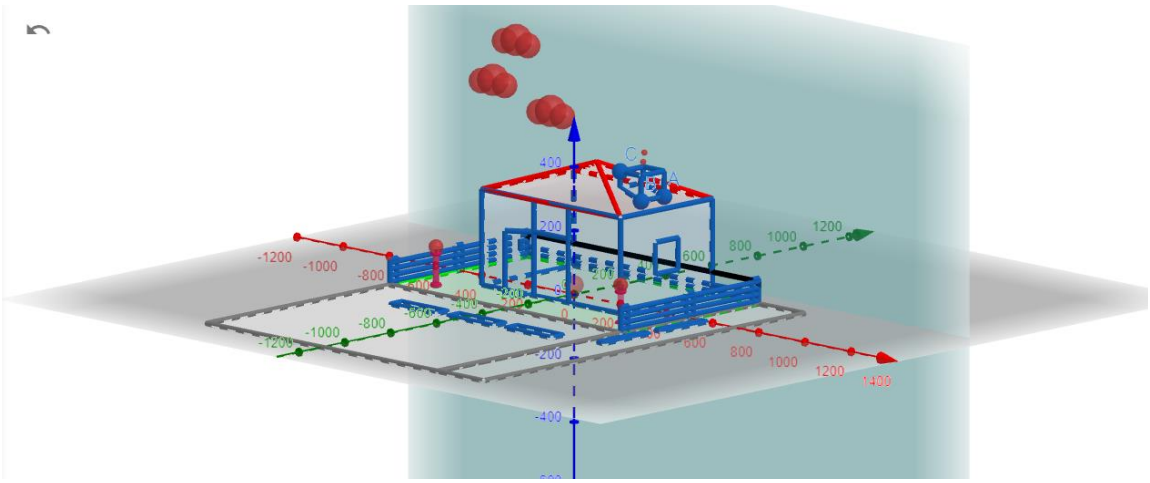
דו"ח מיני פרויקט מבוא להנדסת תוכנה

הפרויקט עוסק בדימוי תמונה ע"י גופים גאומטריים (משלוש, ספירה, פוליגון....)
להלן התמונה הסופית שיצרנו: (בונוס של סיבוב מצלמה נעזרנו בוויקיפדיה)





הסבר מפורט על יצירת התמונה:
שירטוט באתר גאוגברה את התמונה הרצויה ע"י הגופים הנ"ל:
קישור לתמונה שבינינו בגאוגברה – לחץ כאן.



מיני פרויקט שלב 8:

עשינו את השיפור של depth of field, במקום לקחת את נקודת P0 של המצלמה וממנה ליצור קרן דרך פיקסל יצרנו אזור מטרה (grid) מסביב לנקודת P0 ומכל נקודה באזור המטרה בינינו קרן דרך הפיקסל.

פונקציה אשר יוצרת את הנקודות באזור המטרה:

```java
/**
 * Generates a list of points within the target area for super sampling.
 *
 * @param p0            The center point of the target area.
 * @param up            The up vector defining the target area plane.
 * @param right         The right vector defining the target area plane.
 * @param superSampling The density of points per unit distance.
 * @param radius        The radius of the target area.
 * @return The list of points within the target area.
 */
1 usage  נתן שטרן *
public static List<Point> pointsInTheTargetArea(Point p0, Vector up, Vector right, double superSampling, double radius) {
    List<Point> pointsInTarget = new LinkedList<>();
    // Calculate the distance between adjacent points based on the superSampling density and the target area radius.
    double dX = alignZero( number: 2 * radius / superSampling);
    // Iterate over rows and columns to generate points within the target area.
    for (int row = 1; row < superSampling; row++) {
        for (int col = 1; col < superSampling; col++) {
            // Calculate the position of the current point within the target area.
            Point p = p0.add(up.scale( t: dX * row)).add(right.scale( t: dX * col));
            // Add the point to the list of points within the target area.
            pointsInTarget.add(p);
        }
    }
    return pointsInTarget; // Return the list of points within the target area.
}
```

פונקציה אשר יוצרת ריבוי קרניים מאזור המטרה לכיוון נקודה שנמצאת על focal plane

(מישור מדומה):

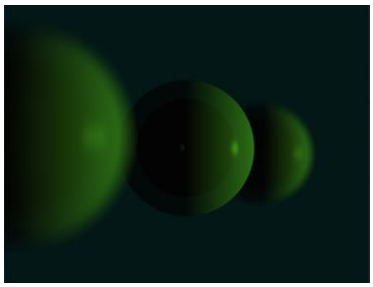
```java
/**
 * Creates a beam of rays from a target area to a specified point.
 *
 * @param points The list of points representing the target area.
 * @param p The destination point.
 * @return A list of rays originating from the points in the target area and pointing towards the destination point.
 */
1 usage  נתן שטרן *
public static List<Ray> createBeamOfRaysFromTargetArea(List<Point> points, Point p) {
    List<Ray> rays = new LinkedList<>(); // Create an empty list to store the rays.
    // Iterate over each point in the list of points representing the target area.
    for (Point point : points) {
        // Create a new ray with the current point as the origin and a direction pointing towards the destination point.
        rays.add(new Ray(point, p.subtract(point)));
    }

    return rays; // Return the list of rays.
```

חישוב ממוצע צבע שנוצר ע"י ריבוי קרניים:

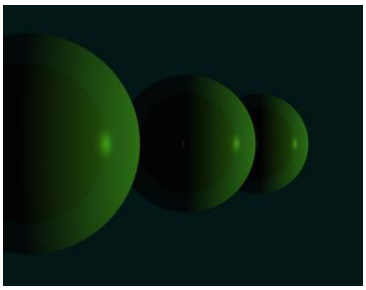
```java
/**
 * Traces multiple rays and returns the average color.
 *
 * @param rays The list of rays to trace.
 * @return The average color of the traced rays.
 */
1 usage  נתן שטרן *
public Color traceMultipleRays(List<Ray> rays) {
    int size = rays.size(); // Get the number of rays in the list.
    Color avgColor = Color.BLACK; // Initialize the average color to black.

    // Iterate over each ray in the list of rays.
    for (Ray ray : rays) {
        // Trace the current ray and add the resulting color to the average color scaled by 1.0 / size.
        avgColor = avgColor.add(traceRay(ray).reduce(size));
    }

    return avgColor; // Return the average color of the traced rays.
}
```

כעת יצרנו תמונה של שלושה כדורים ואנו רוצים לעשות על הכדורים dof:

מצב תמונה לאחר ה – dof ע"י "י תשע קרניים (2^n+1):

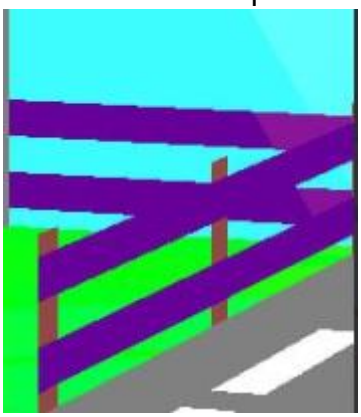


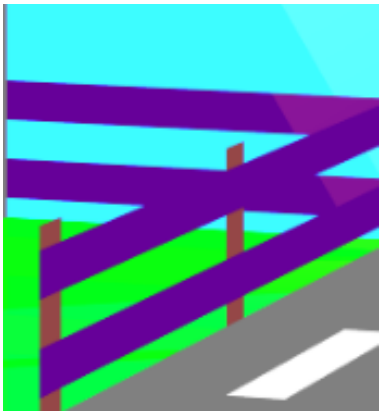
מצב תמונה לאחר ה – dof:



הוספנו גם כן את anti aliasing:

כפי שניתן לראות :





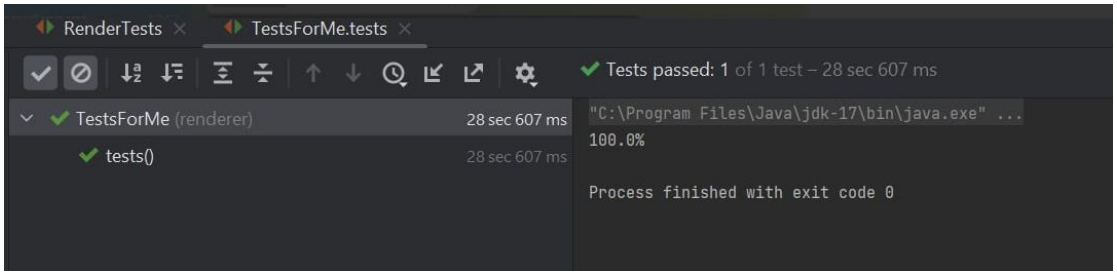
מיני פרויקט שלב 9 :

יש לנו שני שיפורי ביצועים:

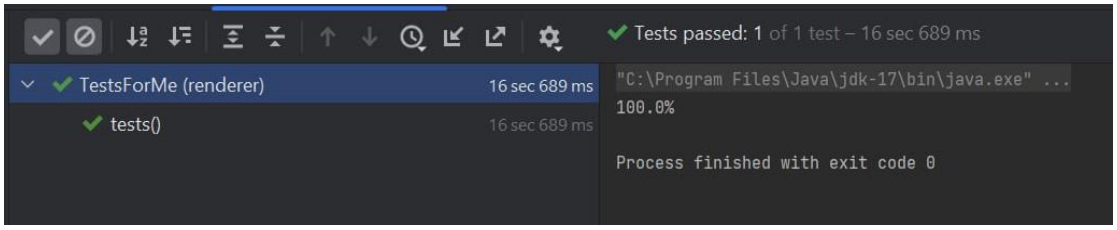
1. ***multi threading***:

במקום שנעשה את זה בתהליך אחד חילקנו את זה לכמה תהליכונים נפרדים.

ללא שיפורים ותהליכונים:



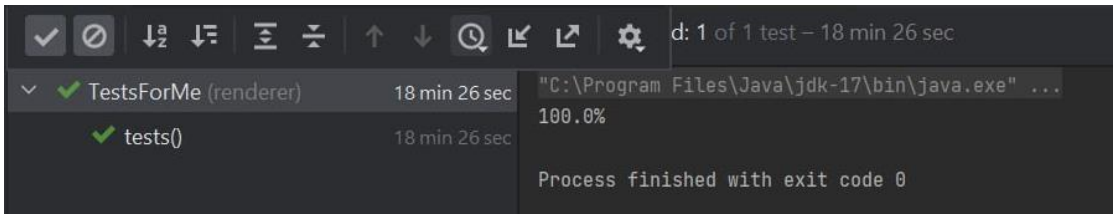
ללא שיפורים עם תהליכונים:



2. ***Adaptive super sampling***:

ניקח כל פיקסל ונחלק אותו לאזורים ולפי זה נקבע את הצבע של הפיקסלים, מה שבעצם קורה שאנחנו בודקים את הצבע במרכז הפיקסל ובארבע קרנותיו,
נבדוק: אם הצבע זהה – נחזיר את הצבע, אחרת ניכנס לפונקציה הרקורסיבית שמחשבת אזור קטן יותר. (ובנוסף נוריד את רמת הרקורסיה)

מצב תמונה של שני השיפורים יחדיו:



(וכמובן שזמן הריצה יהיה הרבה יותר גדול אם לא נשים את שני השיפורים יחדיו).

Bonus:

- טרנספורמציית הזזה וסיבוב. (נקודה 1)
- קבלת מקור תאורה ספוט עם אלומת עור צרה יותר. (נקודה 1)
- השלמת השלב תוך שבוע אחד במקום שבועיים שלב 6. (נקודה 1)
- חיתוך גליל. (נקודה 1)
- הצגת התמונה מהבונוס הנ"ל מזוויות שונות. (נקודה 1)
- מימוש דרך שניה של MaxDistance. (נקודה 1)