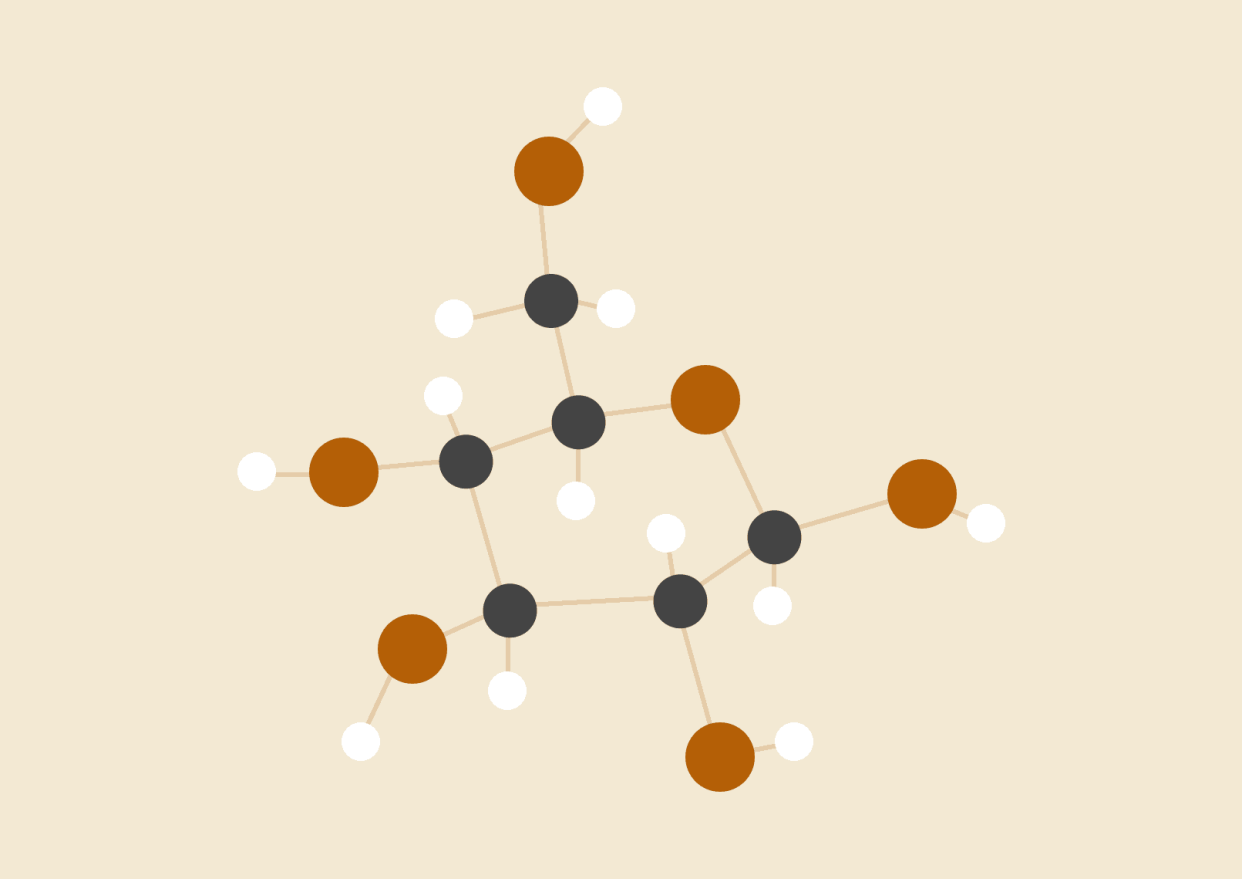
דו"ח מעבדה

"שייבנה בית המקדש שייבנה שוב מחדש" (דודו פישר)



הילה בוזנח-212127435

חנה לאה סילברברג -211416250

תוכן עניינים

Antialiasing

Adaptive SuperSampling

Boundary Volume Hierarchy

Antialiasing

החלקת עקומות - בגלל שעד היום השתמשנו במרכז הפיקסל בלבד לחישוב הצבע שלו, דבר זה יצר לנו קצוות משוננים.

איך נשפר? נעביר כמה קרניים לכל פיקסל, ונעשה ממוצע של הצבעים המוחזרים מהם.

הדלקה וכיבוי של המתודה נעשה ע"י בחירת פונקצייה לרינדור (בחירת renderImageSuperSampling לעומת renderImage)

נשתמש באלגוריתם Super-sampling, אלגוריתם זה ישלח קרניים למקומות רנדומליים בפיקסל, ונרנדר את התמונה לפי ממוצע הצבעים שקרניים אלו יחזירו.

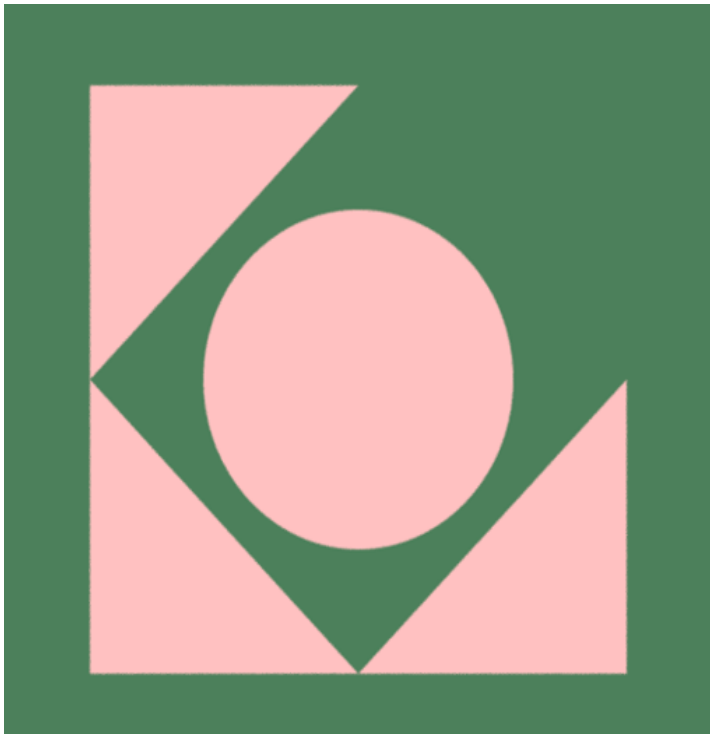
הפונקציה castBeamSuperSampling תשלח קבוצת קרניים (beam) למרכז הפיקסל, ותחשב את הממוצע. ו constructBeamSuperSampling שתיצור קבוצת קרניים רנדומלית בהן נשתמש.

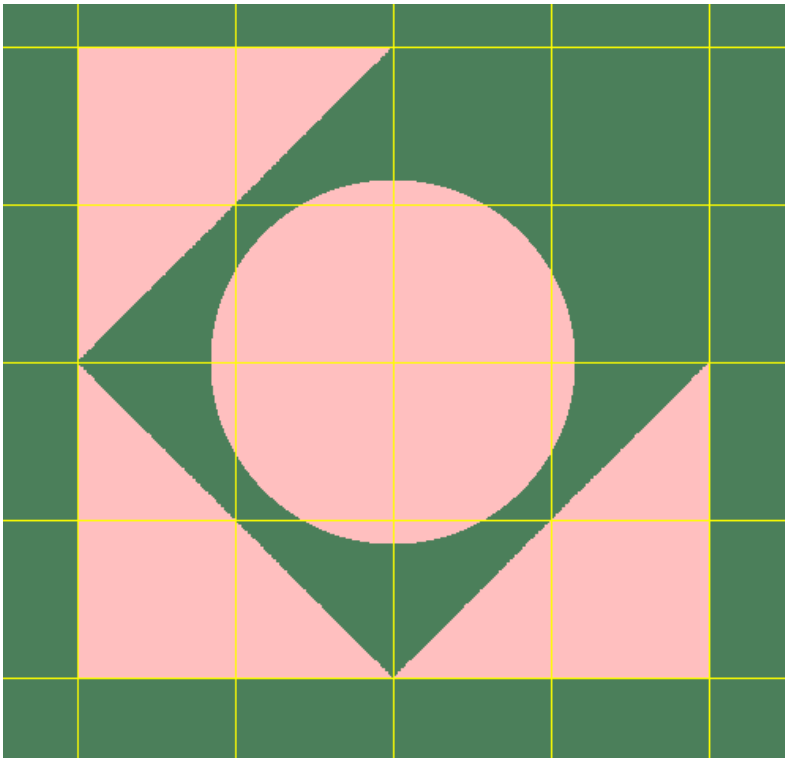
```
private List<Ray> constructBeamSuperSampling(int nX, int nY, int j, int i) {
    List<Ray> beam = new LinkedList<>();
    beam.add(constructRay(nX, nY, j, i));
    double ry = height / nY;
    double rx = width / nX;
    double yScale = alignZero( number: (j - nX / 2d) * rx + rx / 2d);
    double xScale = alignZero( number: (i - nY / 2d) * ry + ry / 2d);
    Point pixelCenter = p0.add(vTo.scale(distance)); // center
    if (!isZero(yScale))
        pixelCenter = pixelCenter.add(vRight.scale(yScale));
    if (!isZero(xScale))
        pixelCenter = pixelCenter.add(vUp.scale( scaleFactor: -1 * xScale));
    Random rand = new Random();
    // create rays randomly around the center ray
    for (int c = 0; c < nSS; c++) {
        // move randomly in the pixel
        double dxfactor = rand.nextBoolean() ? rand.nextDouble() : -1 *
                rand.nextDouble();
        double dyfactor = rand.nextBoolean() ? rand.nextDouble() : -1 *
                rand.nextDouble();
        double dx = rx * dxfactor;
        double dy = ry * dyfactor;
        Point randomPoint = pixelCenter;
        if (!isZero(dx))
            randomPoint = randomPoint.add(vRight.scale(dx));
        if (!isZero(dy))
            randomPoint = randomPoint.add(vUp.scale( scaleFactor: -1 * dy));
        beam.add(new Ray(p0, randomPoint.subtract(p0)));
    }
```

```
1 usage   Administrator
private Color castBeamSuperSampling(int j, int i) {
    List<Ray> beam = constructBeamSuperSampling(imageWriter.getNx(), imageWriter.getNy(), j, i);
    Color color = Color.BLACK;
    // calculates the average colour of rays traced
    for (Ray ray : beam) {
        color = color.add(rayTracer.traceRay(ray));
    }
    return color.reduce(nSS);
}
```

לפני ואחרי:

ניתן לראות כי השיפור בהחלט "מחליק" את הקצוות המשוננים.





Adaptive SuperSampling

לעיתים קרובות, ובייחוד עם השיפור הקודם , שיפור ה -antialiasing (ובמיוחד אצלינו בתמונה הסופית), נשלח קרניים רבות לאזורים בהם יש צבע אחיד, ויקרה שנשלח קרניים רבות לחשב אותו צבע שוב ושוב (סוג של needless complexity).

איך נשפר? נחלק את הפיקסל (דגימה) ל 4 חלקים קטנים יותר, נשווה בין הצבעים של החלקים, אם הצבעים אותו דבר - נצא, אין מה להמשיך בשליחת קרניים לאזורים בעלי צבע אחיד, אחרת - נפצל וכך נציג דיוק רב יותר בצבע.

Boundary Volume Hierarchy

כל צורה, ניצור אזור תוחם, שמרני, שיהיה מהר מאוד לבדוק האם הקרן חותכת אותו או לא. וכך לא נכנס לכל צורה את החיתוכים שלה. אלא רק אם האזור התוחם נחתך. האם הקרן חותכת את הצורה שלנו בכלל.