דו"ח פרויקט

(מיני פרויקט 1)

- כל הפונקציות מחזירות פרמטרים חדשים תמיד – כולל הפונקציות של חיבור/חיסור, או הכפלה
- וקטורית (למעט נרמול שמתבצע על האיבר עצמו), בהתאם להנחיות רכז הקורס.
- השתמשנו כמעט תמיד בשמות משתנים אינדיקטיביים למעט במקומות בהם העתקנו נוסחאות מתמטיות, במקרה כזה, העתקנו את שמות המשתנים מהנוסחה עצמה.
- הסברים לפני כל פונקציה, ובמקרה הצורך גם in-code כדי לאפשר קריאה נוחה וקלה של הקוד גם למי שקורא אותו לראשונה.
- Javadoc - הסבר פורמלי עבור כל מחלקה.

השיפורים:

הוספנו שני שיפורים למנוע שלנו:

1. Depth of Field ליצירת אפקט עומק (Bokeh) בתמונות, כפי שניתן לראות בתמונה הזאת:

לפני השיפור, כל התמונה בפוקוס, מראה לא טבעי

לאחר השיפור, האובייקטים הקרובים למצלמה והאובייקטים הרחוקים ממנה נראים מטושטשים, כפי שהם נראים בצילום במצלמה אמיתית

על מנת להפעיל את השיפור, יש להפעיל במצלמה המוגדרת בטסט את השורות הבאות:

- קביעת הפעלה של האפקט
- קביעת מרחק מישור הפוקוס
- קביעת גודל הצמצם
- קביעת מספר הקרניים בתוך הצמצם

הדגמה של הפעלת האפקט בעמוד הבא

```
Camera camera = new Camera(
        new Point3D( x: 30,  y: 150,  z: 20),
        new Vector( x: -5,  y: -27,  z: -3),
        new Vector( x: -1.2,  y: -2,  z: 20))
        .set_DOF(true)
        .setFocalDistance(140)
        .setApertureSize(100)
        .setNumberOfRaysInAperture(81)
```

בתמונה המשופרת למעלה קבענו:

את אפקט העומק להיות true.

את מרחק משטח הפוקוס (ה־focal plane) ל־140.

את גודל הצמצם ל־100.

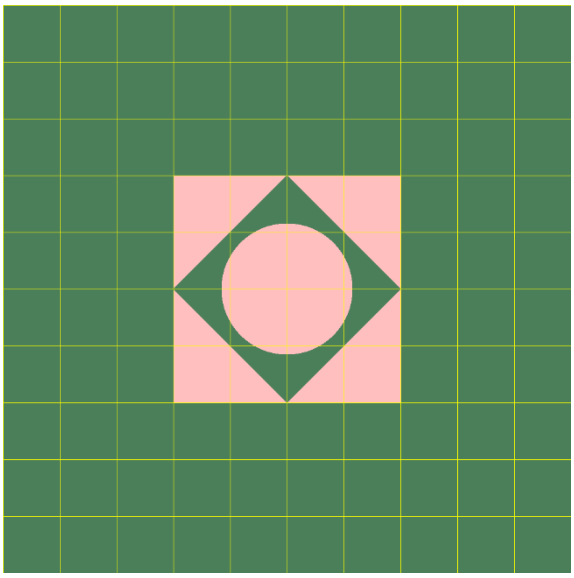
את מספר הקרניים לכל פיקסל ל־81.

הערה:

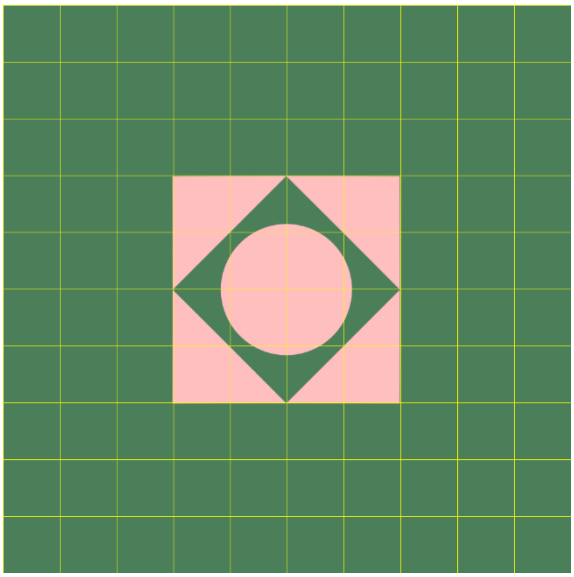
*פיזור הכוכבים מתבצע רנדומלית בכל הרצה, ולכן משתנה מתמונה לתמונה

2. שיפור Anti-Aliasing ליצירת מראה טבעי ומוחלק יותר בקצוות של אובייקטים המופיעים בתמונות, כפי שניתן לראות בהדגמה הפשוטה הזאת:

לפני השיפור, הקצוות של הכדור נראים משוננים בצורה לא טבעית



לאחר השיפור, קצוות הכדור מוחלקים ונראים טבעיים יותר



על מנת להפעיל את השיפור, יש להפעיל במצלמה המוגדרת בטסט את השורות הבאות:

- קביעת הפעלה של האפקט
- קביעת מספר הקרניים לכל פיקסל

הדגמה של הפעלת האפקט בעמוד הבא

```
camera
        .setAA(true)
        .setNumberOfRaysInPixel(36);
```

בתמונה המשופרת למעלה קבענו:

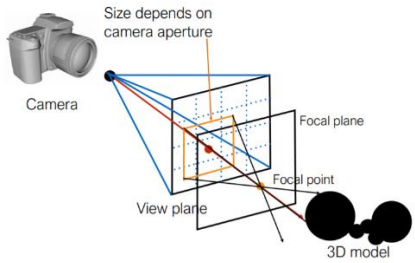
את אפקט ה־AA להיות true.

את מספר הקרניים לכל פיקסל ל־36.

הערה:

*מומלץ להגדיל את התמונות כדי לראות ביתר בירור את השפעת האפקט.

הסבר קצר על מימוש השיפורים – Depth of Field:



מתבצע ע"י יצירת צמצם (במקרה שלנו בחרנו לעשות אותו עגול כדי להשיג אפקט טוב יותר) והעברתו על פני כל פיקסל בנפרד, כפי שנראה במצגת ההסבר על השיפורים.

במקום להשתמש בפונקציה הזו שמטילה **קרן בודדת**

```
private void castRay(int nX, int nY, int col, int row) {
```

שקוראת לפונקציה

```
public Ray constructRayThroughPixel(int nX, int nY, int j, int i) {
```

השתמשנו בפונקציה שמטילה **אלומה**

```
private void castBeam(int nX, int nY, int col, int row) {
```

שקוראת לפונקציה שמעבירה **יותר מקרן אחת** לכל פיקסל

```
public List<Ray> constructRaysThroughPixel(int nX, int nY, int j, int i) {
```

בתוך הפונקציה הזו, אם המשתנה שמגדיר כמות קרניים לצמצם אינו 1

```
if (_numberOfRaysInAperture != 1) {
```

ניצור רשימת קרניים מפוזרות בצורה אחידה בשטח הצמצם בעזרת הפונקציה הזו

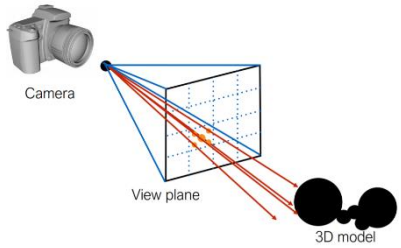
```
public List<Ray> randomRaysInCircle(
```

ונחזיר את רשימת הקרניים, נעקוב אחרי כל קרן, נחשב את ממוצע הצבעים עבור הפיקסל, ונכתוב אותו בתמונה, באופן הבא

```
List<Ray> rays = camera.constructRaysThroughPixel(nX, nY, col, row);

List<Color> colors = new LinkedList<>();
for (Ray ray : rays) {
    colors.add(tracer.traceRay(ray));
}
imageWriter.writePixel(col, row, Color.avgColor(colors));
```

הסבר קצר על מימוש השיפורים – Anti-Aliasing:



מתבצע ע"י שליחת קרניים מרובות המפוזרות ברחבי הפיקסל, וחישוב הממוצע עבור כל הקרניים.

במקום להשתמש בפונקציה הזו שמטילה **קרן בודדת**

```
private void castRay(int nX, int nY, int col, int row) {
```

שקוראת לפונקציה

```
public Ray constructRayThroughPixel(int nX, int nY, int j, int i) {
```

השתמשנו בפונקציה שמטילה **אלומה**

```
private void castBeam(int nX, int nY, int col, int row) {
```

שקוראת לפונקציה שמעבירה **יותר מקרן אחת** לכל פיקסל

```
public List<Ray> constructRaysThroughPixel(int nX, int nY, int j, int i) {
```

בתוך הפונקציה הזו, אם המשתנה שמגדיר כמות קרניים לפיקסל אינו 1

```
if (_numberOfRaysInPixel != 1)
```

ניצור רשימת קרניים מפוזרות בצורה אחידה בשטח הצמצם בעזרת הפונקציה הזו

```
public List<Ray> randomRaysInGrid
```

ונחזיר את רשימת הקרניים, נעקוב אחרי כל קרן, נחשב את ממוצע הצבעים עבור הפיקסל, ונכתוב אותו בתמונה, באופן הבא

```
List<Ray> rays = camera.constructRaysThroughPixel(nX, nY, col, row);

List<Color> colors = new LinkedList<>();
for (Ray ray : rays) {
    colors.add(tracer.traceRay(ray));
}
imageWriter.writePixel(col, row, Color.avgColor(colors));
```