דו"ח פרויקט  
יעל תפארת צ'יקו ואביגיל כהן

כללי:

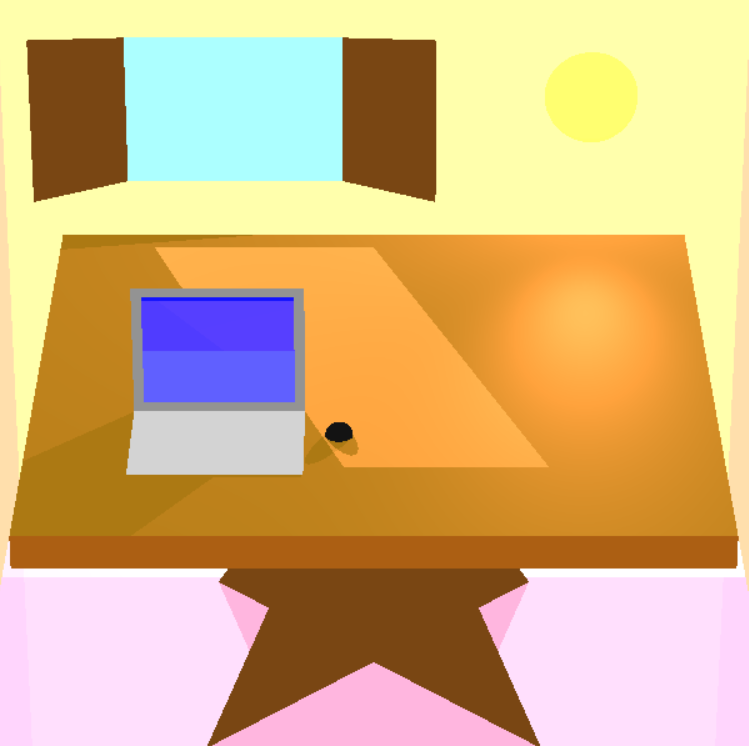
-כתיבת כל המחלקות והפונקציות הנדרשות על פי ההנחיות, ובמידת הצורך הוספת פונקציות עזר פרטיות.

-מימוש כל הטסטים הנדרשים, חלקם בעזרת הקוד שניתנו לנו במודל, וחלקם כתבנו לבדנו.

-יצירת setters במקום הנדרש, במקרה הצורך החזרת האובייקט, בתבנית builder. רק איפה שנצרך יצרנו getter.

-הסברים בכל פונקציה מה תפקידה ע"פ פורמט Java Doc. הסבר עבור כל מחלקה, שדה, משתנה וכל פונקציה בצורה פורמאלית ונהירה.

-תמונה עבור שלב ובונוס שלב 7: תמונה מורכבת, בה משולבים כל סוגי הצורות והאורות, כמו כן באפקטים שקיפות השתקפות.



שיפורים:

**-שיפור מיני פרויקט 1: depth of field (עומק שדה):**

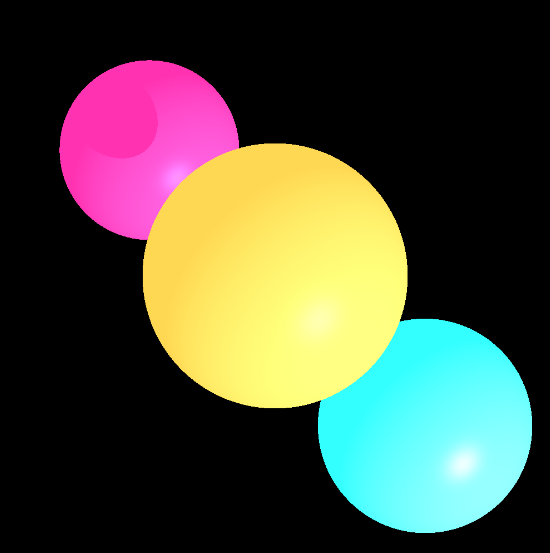
שיפור זה מטרתו ליצור כמין פוקוס על כל הגופים הנמצאים במרחק ספציפי (תלוי במשתמש) מהמצלמה.  
עבור כל פיקסל על ה viewPlane הוצאנו את הקרן הראשית ממרכז הצמצם לעבר מרכז הפיקסל. בדקנו היכן הקרן חותכת את משטח הפוקוס- ה"פוקל פליין" focal plane)).  
עבור כל נקודה על הצמצם הוצאנו קרן ממנה לעבר נקודת החיתוך שמצאנו.

מחלקת Camera:  
כדי שתהיה אפשרות בחירה האם להשתמש בשיפור הוספנו פונקציה חדשה בנוסף לפונקציה המקורית הבונה קרן בודדת עבור כל פיקסל. הפונקציה החדשה בונה כמה קרניים במקום אחת.  
הפונקציה המקורית: constructRayThroughPixel, הפונקציה החדשה: constructRaysThroughPixel.   
כדי שיצאו כמה קרניים עבור כל פיקסל, הוספנו פונקציה ושדות בהתאמה.   
השדות:

:apertureSizeגודל הצמצם.   
:aperturePoints רשימת הנקודות על הצמצם.   
apertureN: כמות פיקסלים (עבור רוחב וגובה) בצמצם.   
focalDist: מרחק בין המצלמה והפוקל פליין.

הפונקציה: פונקציה בשם aperturePointsInit שתפקידה למצוא את כל הנקודות על הצמצם. הפונקציה עוברת על משטח הצמצם ובעזרת וקטורי כיוון (למעלה וימינה) מוצאת את כל מרכזי משבצות רשת הצמצם ומחזירה אותן כרשימת נקודות.

מחלקת Render:   
כפי שהסברנו צריך שתהיה אפשרות בחירה האם להשתמש בשיפור ולכן הוספנו בנוסף לפונקציה renderImage הבונה את התמונה, את הפונקציה renderFocusImage שתבנה את התמונה עם פוקוס.  
הפונקציה די דומה לפונקציה המקורית אך במקום לצבוע את הפיקסל בצבע של הגוף בו פוגעת הקרן הבודדת, היא צובעת את הפיקסל בממוצע הצבעים של כל הנקודות בהן פוגעות הקרניים המרובות בפיקסל הבודד (הקרניים שקיבלנו מהפונקציה constructRaysThroughPixel שבמחלקת Camera). בצורה זו הצבע במישור הפוקוס יהיה חד יותר בכל פיקסל- והפיקסל אוטומטית ייראה יותר מפוקס.

הטסט שכתבנו כדי לבחון את השיפור מורכב מ3 עיגולים- כל אחד במרחק אחר מהמצלמה. ימין (כחול): נמצא על הפוקל פליין- עליו מכוון הפוקוס.  
אמצע (צהוב): במרחק 250 לפני הפוקל פליין.  
שמאל (ורוד): במרחק 250 מאחורי הפוקל פליין.

**-שיפור מיני פרויקט 2: adaptive super sampling**

שיפור זה מטרתו לחסוך בזמן ריצה ע"י שליחת כמות מועטה יותר של קרניים. נעשה זאת בעזרת חלוקת הצמצם לחלקים קטנים יותר ויותר ושליחת קרניים בהתאם לצבעים בפינות ובמרכז שטח החלוקה. יש לציין שהשיפור התבצע בהתאמה לשלב ראשון של מיני פרוייקט- depth of field.

מחלקת Camera:  
קודם מצאנו את כל הנקודות על הצמצם ע"י פונקציית aperturePointsInit, ושמרנו אותן במטריצת נקודות, כאשר כל נקודה נמצאת במקום המתאים לאינדקסי x וy ע"פ המיקום שלה בצמצם (קודם x ואז y). את המטריצה שמרנו כשדה pointsMat, והוספנו פונקצייה getPointByMat שתחזיר את הנקודה על הצמצם ע"פ המיקום שלה במטריצה.   
את אתחול המטריצה והקריאה לפונקצייה aperturePointsInit עשינו בפונקצייה setAperture.  
בנוסף על כך ביצענו רפרקטורינג והעברנו את קטע הקוד שמוצא את הנק' על הפוקל פליין עבור פיקסל מסוים (מהשלב הקודם) לפונקצייה בשם getFocusPoint שתחשב את הנקודה.

מחלקת Render:   
קודם כל הוספנו 2 שדות בוליאניים המאותחלים כ false: isMultyRay עבור הפעלת depth of field, ו isAdaptiveSS עבור הפעלת adaptive super sampling. הפעלת פונקציית setMultyRay הופכת את isMultyRay ל true. הפונקצייה גם מקבלת ערך בוליאני לפיו היא מחליטה אם להפעיל את isAdaptiveSS.  
בהתאמה לכך בפונקציות renderImage ו renderImageThreaded הוספנו בדיקה שבודקת את השדות הבוליאנים ועל פיהם קוראת ל castRay המתאים (עבור קרן בודדת \ כל הקרניים \ רק חלק מהקרניים).

הפונקציות שאחראיות לחלוקת הצמצם ולשליחת הקרניים המתאימות הן castRaysASS ו castRaysASSInner, פונקציית מעטפת ופונקציה רקורסיבית פנימית.

castRaysASS: פונקציה האחראית על צביעת פיקסל. בפונקציה זו מאותחלת מטריצת צבעים, בה ישמר עבור הפיקסל כל צבע מנק' על הצמצם. המיקום במטריצה תואם למקום הנק' במטריצת הנקודות על הצמצם. לאחר מכן צבע הפיקסל יחושב ע"י קריאה לפונקצית הרקורסיה.

castRaysASSInner: הפונקציה מקבלת את השדות הבאים:

focusPoint: הנק' על הפוקל פליין  
colors: מטריצת הצבעים של הפיקסל  
x1: שיעור x במטריצה, של הפינה השמאלית למטה  
y1: שיעור y במטריצה, של הפינה השמאלית למטה  
x2: שיעור x במטריצה, של הפינה הימנית למעלה  
y2: שיעור y במטריצה, של הפינה הימנית למעלה

הפונקציה תחשב את הצבע בארבעת הפינות ע"י שימוש בפונקציית העזר averageColor4. אם הגענו לחלוקה האחרונה- נחזיר צבע ממוצע של הפינות.  
אחרת נשלח כל רבע לפונקציה הרקורסיבית עצמה ונחזיר את ממוצע הצבעים שיתקבלו מכל הרבעים.

averageColor4: פונקציית עזר שמחשבת צבעים ב4 פינות ובנק' האמצע. הפונקציה מקבלת את אותן הפרמטרים של castRaysASSInner.  
קודם כל נחשב את שיעורי הx והy של 2 הפינות שלא קיבלנו, ובעזרת האינדקסים נקבל את הנקודות המתאימות על הצמצם. לאחר מכן נבדוק עבור כל פינה אם הצבע שלה כבר קיים במטריצת הצבעים. אם כן- נקבל אותו משם, ואם לא נחשב אותו בעזרת ה ray Tracer.  
בסוף נמצא את נקודת האמצע ע"י מיקומי הנקודות של הפינות במרחב, ונבדוק האם יש 2 נקודות מתוך ה5 שהפרש צבעיהן גדול מהפרש שהגדרנו כשדה DELTA. אם כן נחזיר Null, ואם לא נחזיר את ממוצע 5 הצבעים.

להלן זמני הריצה עבור הטסט משלב קודם:  
בלי פוקוס, בלי adaptiveSS, בלי threads: 1.5 שניות  
בלי פוקוס, בלי adaptiveSS, עם threads: 1.2 שניות  
עם פוקוס, בלי adaptiveSS, בלי threads: 36.5 שניות  
עם פוקוס, בלי adaptiveSS, עם threads: 20.7 שניות  
עם פוקוס, עם adaptiveSS, בלי threads: 27.9 שניות  
עם פוקוס, עם adaptiveSS, עם threads: 14.4 שניות

**-תמונה סופית**

התמונה הסופית מורכבת ממישור, מצולעים, משולשים וספירות. בתמונה ניתן לראות מגדלור בים בלילה, על חצי אי, וירח וככבים ברקע.   
בקצה המגדלור ישנם 4 מקורות אור point light, והשמשות של החלון (בצהוב) בעלות שקיפות מסוימת בהתאם כדי שהאור יצא.   
הים מורכב ממישור בעל מעט השתקפות.

זמני הריצה:  
בלי פוקוס, בלי adaptiveSS, בלי threads: 18 שניות  
בלי פוקוס, בלי adaptiveSS, עם threads: 15 שניות  
עם פוקוס, בלי adaptiveSS, בלי threads: 90 דקות  
עם פוקוס, בלי adaptiveSS, עם threads: 55 דקות  
עם פוקוס, עם adaptiveSS, בלי threads: 10 דקות  
עם פוקוס, עם adaptiveSS, עם threads: 7.5 דקות

נתוני הפוקוס:   
פוקוס על האי והמגדלור. גודל צמצם: 16, N=4— 2^4 משבצות בשורה/ טור.



ללא פוקוס



עם פוקוס