**דו"ח שיפורים**

**מיני פרויקט במבוא להנדסת תוכנה**



**מרצה:**

אליעזר גינזבורגר

**מגישים:**

ינון חיון 204491609

נדב צימרמן 318591997

# תוכן עניינים

**יצירת התמונה3**

**שיפורי תמונה4**

Anti-aliasing4

Soft shadow6

**שיפורי ביצועים8**

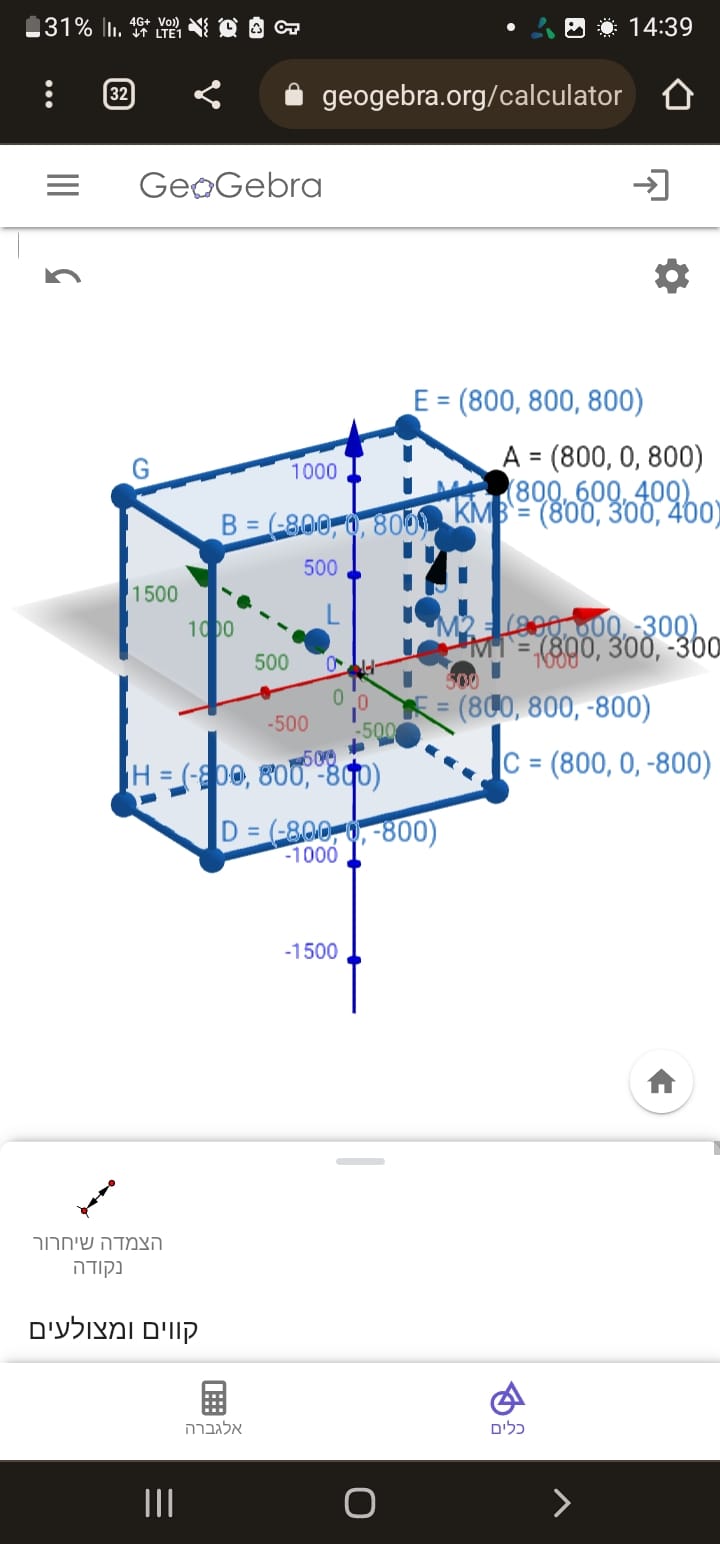
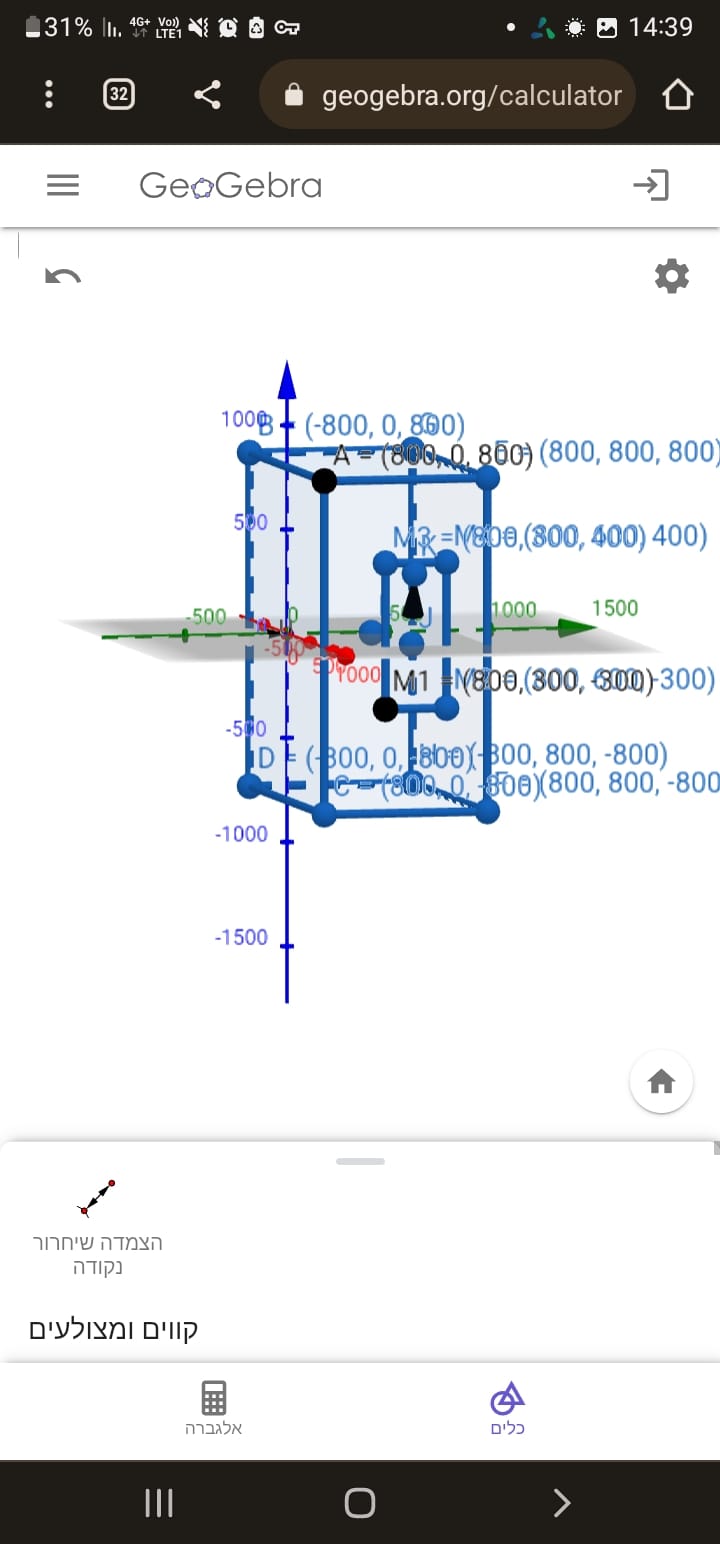
BVH8

Multi-Threading 11

**בונוסים12**

# שלבי יצירת התמונה

את התמונה הסופית יצרנו על בסיס סקיצה ראשונית שנעשתה על גרף תלת מימדי בעזרת התוכנה GeoGebra.

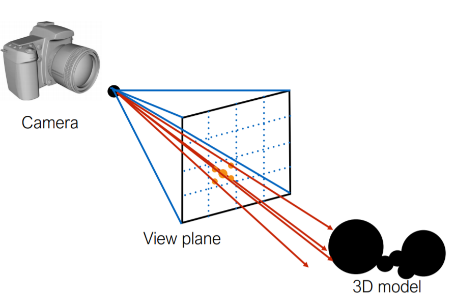
מיקמנו את החדר במרכז ראשית הצירים כאשר המצלמה מגיעה מכיוון השלילי של ציר הY.[[1]](#footnote-2)

לאחר מכן יצרנו את התמונה ב IntelliJ ולאחר מכן הוספנו ריהוט וחפצים לחדר (מיטות, כריות, שידות, מראות, שעון, ארון קיר)

כמו כן הוספנו בחדר מקורות אור שונים

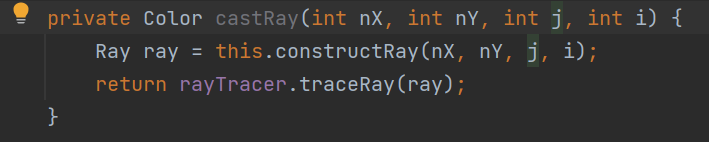
* אור שמש (ישיר) שנכנס דרך החלון
* Point light (על הקיר השמאלי)
* Spot light (בתקרת החדר)
* 2 Spot light (מעל המיטה)

# שיפורי תמונה

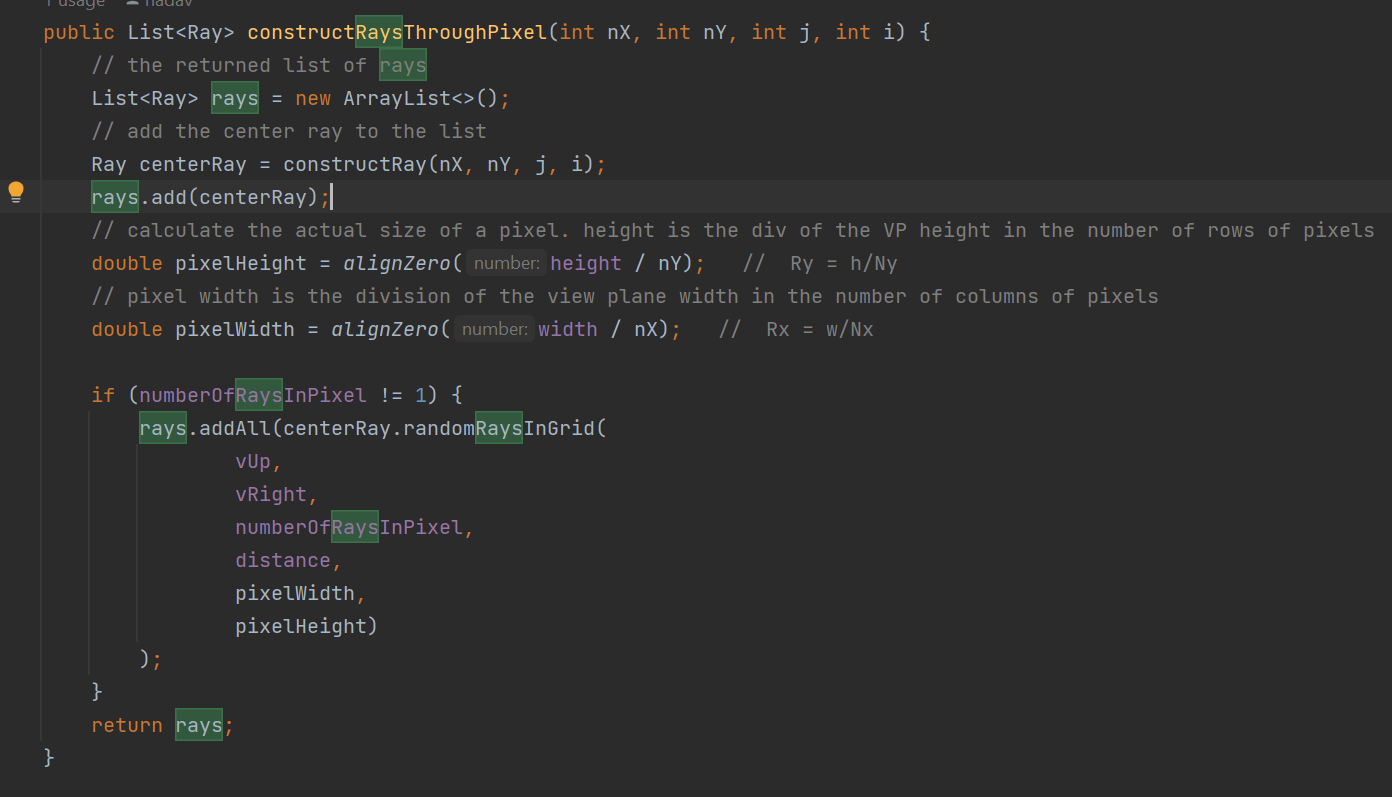
, AA Anti-aliasing:

שיפור זה נועד ליצירת מראה טבעי וחלק יותר בקצוות של אובייקטים המופיעים בתמונות, כפי שניתן לראות בהדגמה הפשוטה הזאת. השיפור בעקבות לקחת דגימות מרובות של הצבע בכל נקודה. נזכיר כי הצבע נקבע על ידי החיתוך של הקרן המשוגרת מן הview palne עם צורות התמונה. בשיפור זה אנו יורים מכל פיקסל מספר רב של קרניים, סוכמים את הצבע שהתקבל מכל הקרניים שנורו בפיקסל וזהו הצבע הסופי שיתקבל בנקודה זו של התמונה.

**אופן מימוש השיפור:**

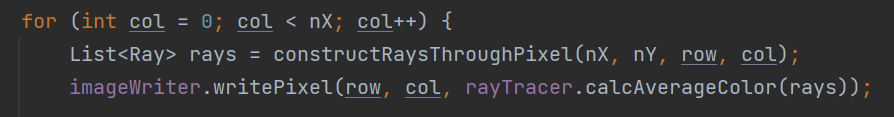
במקום להשתמש בפונקציה הזו שמטילה קרן בודדת:

שקוראת לפונקציה שמשגרת קרן דרך הפיקסל:

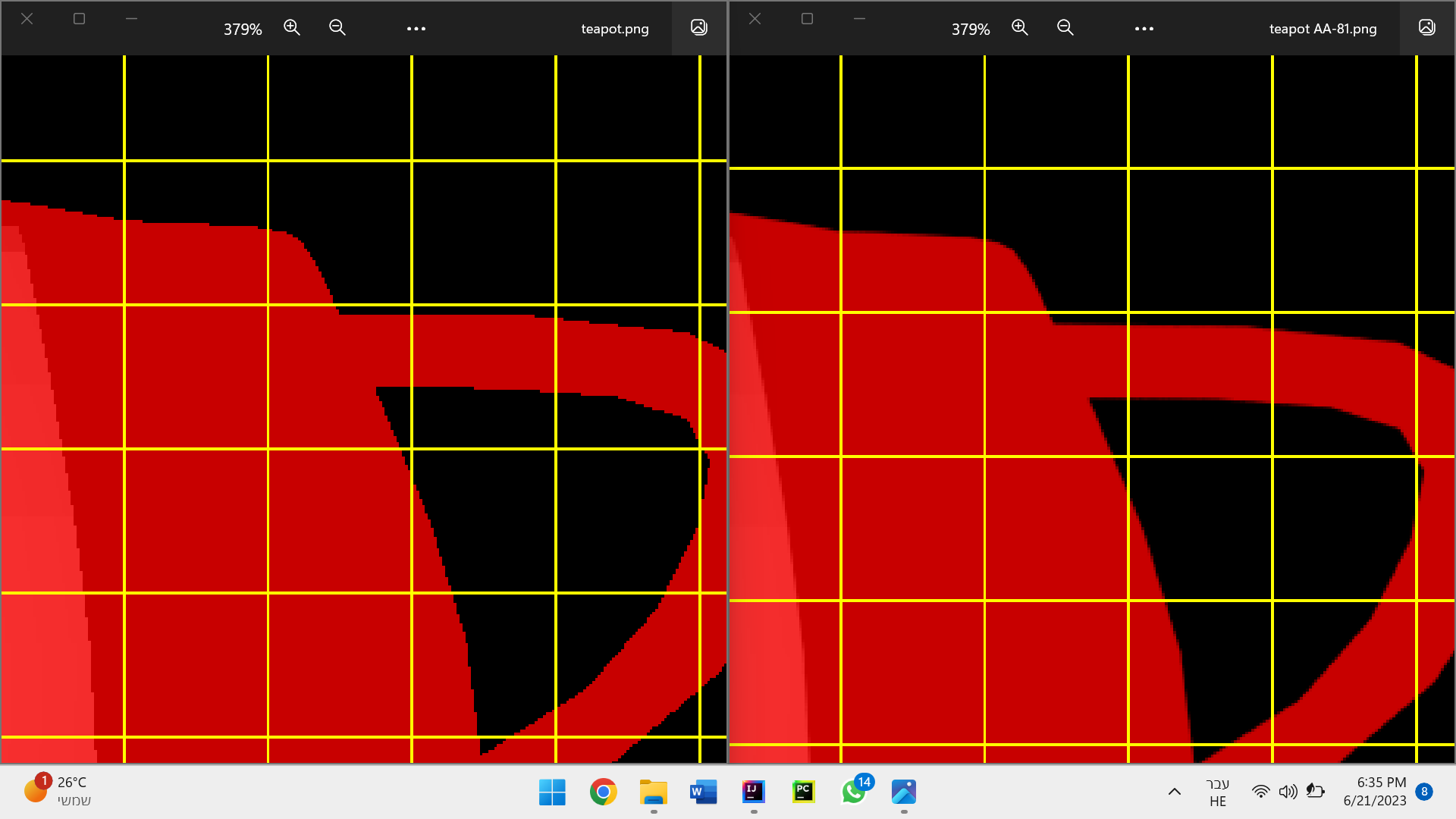
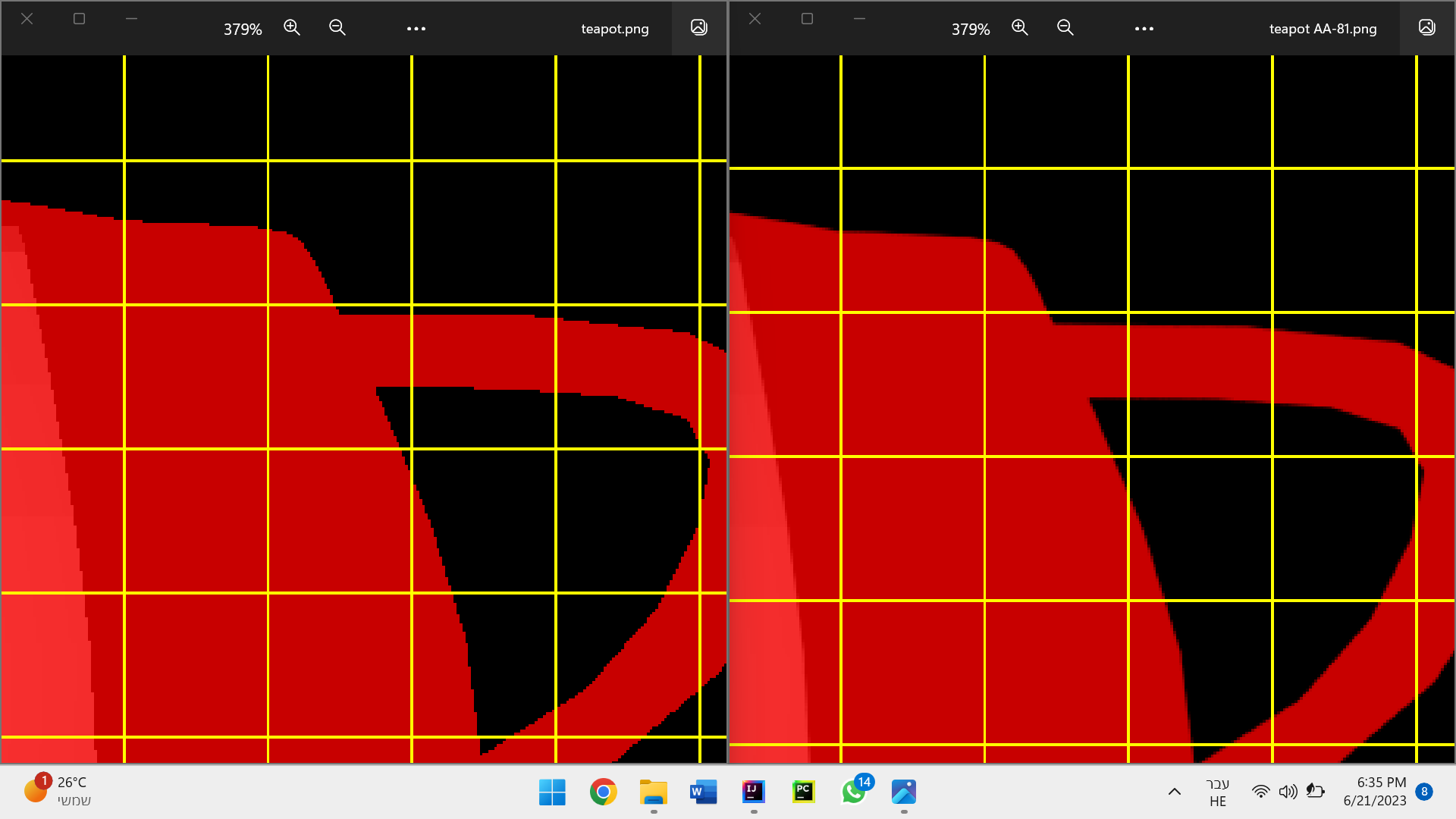
השתמשנו בפונקציה שיוצרת אלומת קרניים בשיטה רנדומלית יחד עם שימוש בגריד:

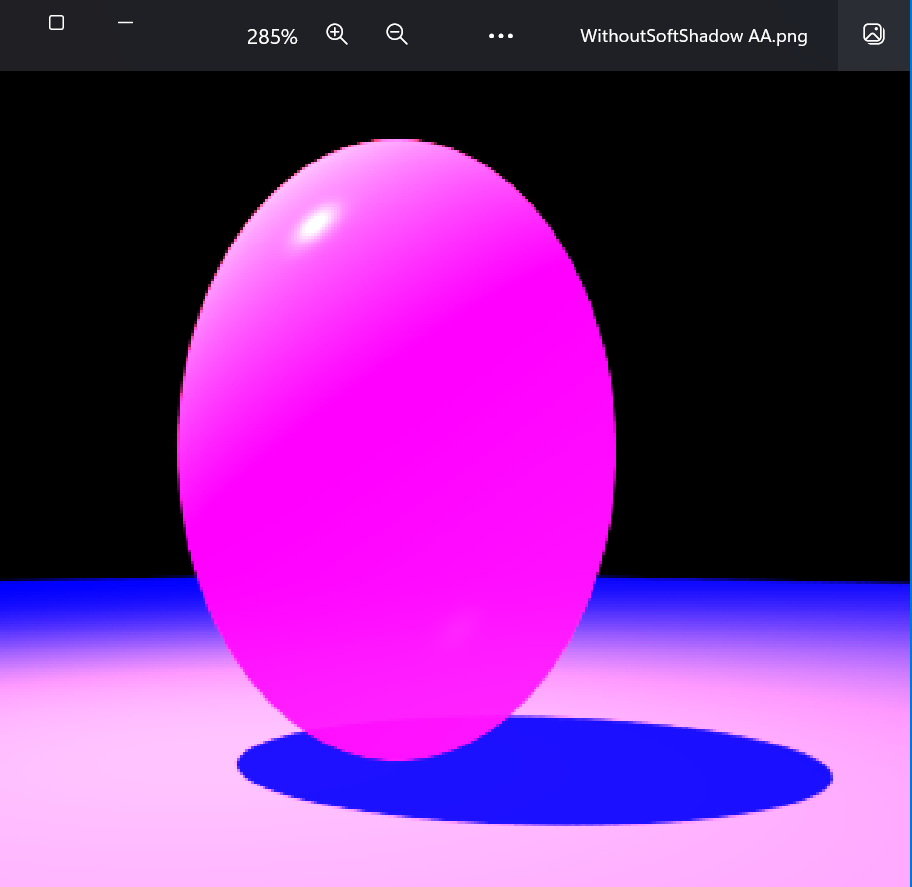
בתוך הפונקציה הזו, אם המשתנה שמגדיר כמות קרניים שונה מאחד ניצור רשימת קרניים מפוזרות בצורה אחידה בשטח הפיקסל בעזרת הפונקציה randomRaysInGrid:

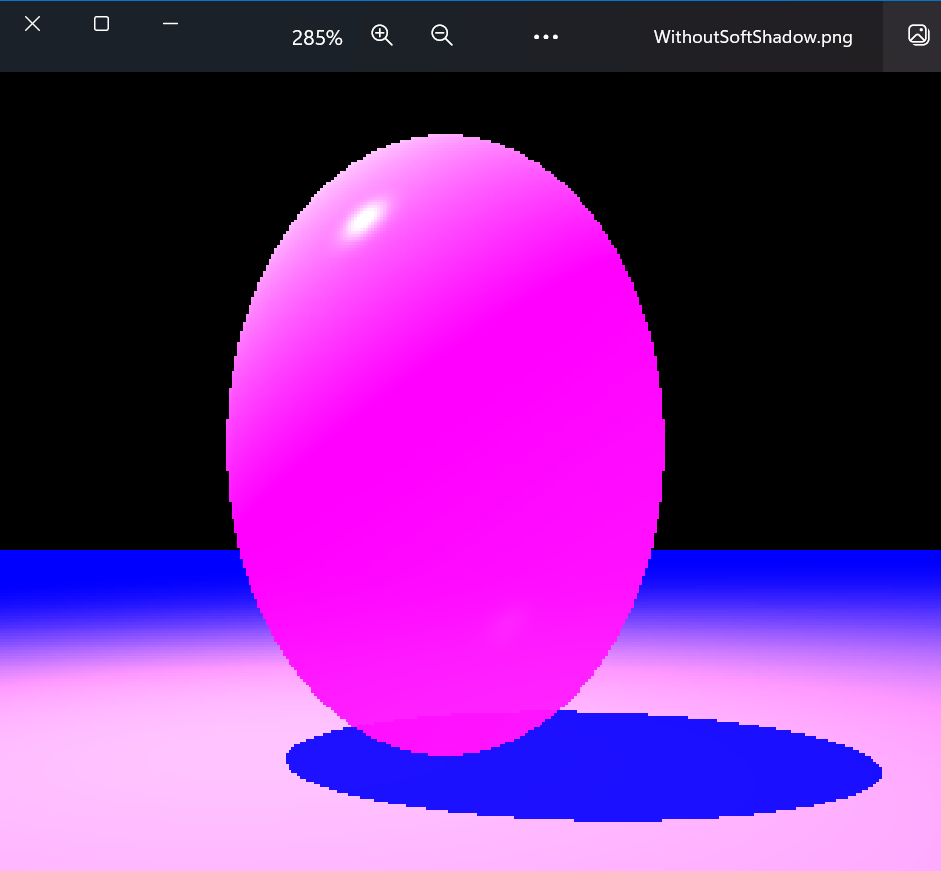


בסופו של דבר נחזיר את רשימת הקרניים, נשגר כל קרן, נחשב את ממוצע הצבעים עבור הפיקסל, ונכתוב אותו בתמונה, באופן הבא:

**הדגמת השיפור:**

הקומקום לפני שיפור Anti-aliasing הקומקום לאחר שיפור Anti-aliasing

כדור לפני שיפור Anti-aliasing כדור לאחר שיפור Anti-aliasing



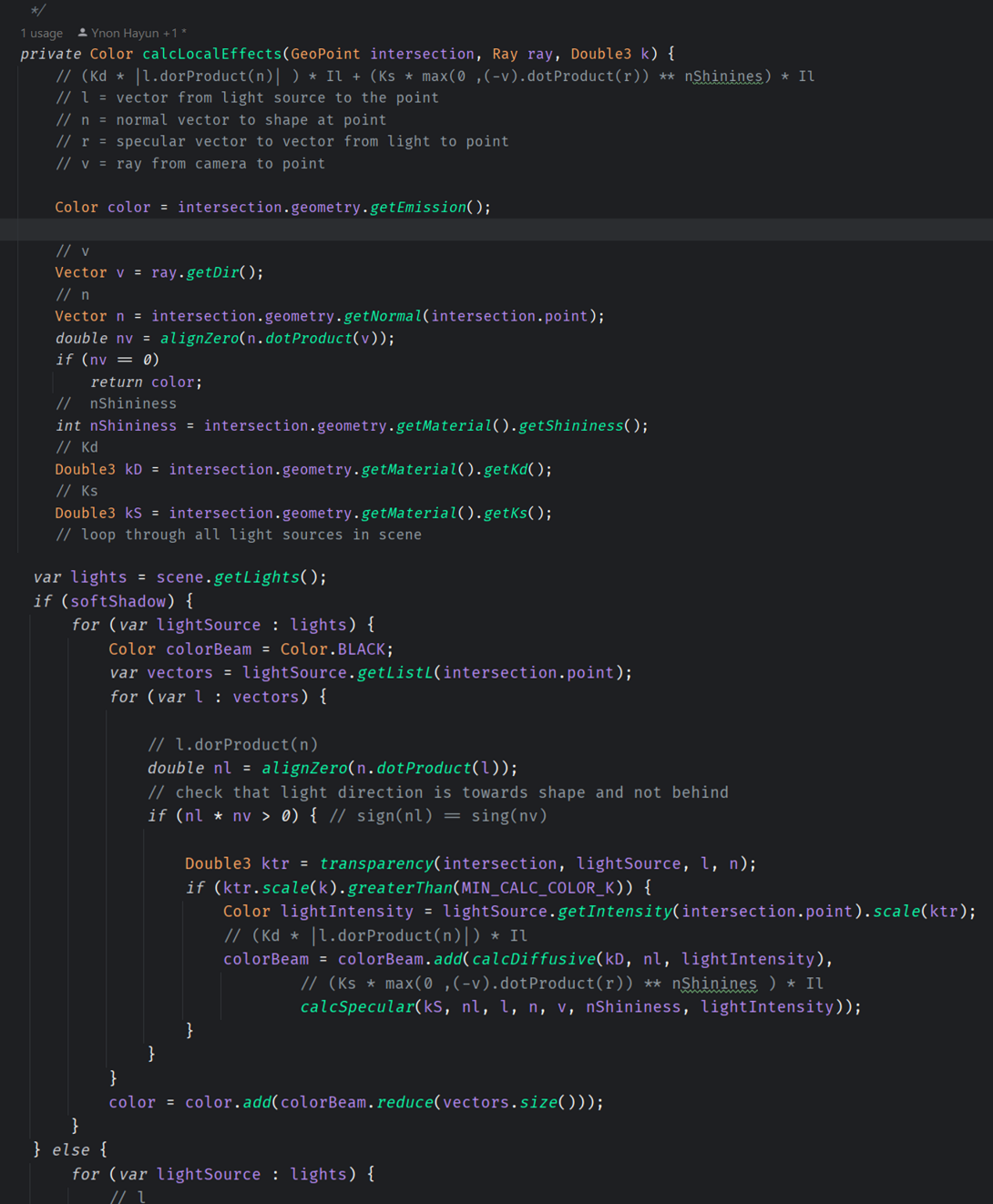
Soft shadow:

בעיה נוספת ששמנו לב היא שאם נסתכל מקרוב על הצללים בסצנה, אנו שמים לב שהם מאוד "קשים". כלומר, שיש לנו אזור שמוצל לגמרי, ואז ממש לידו יש לנו אזור לא מוצל עם קו ברור מאוד ביניהם, מה שאינו דומה לצל בעולם האמיתי, כשהצל דוהה לאט עד לאזור שאינו מוצל לחלוטין.

כדי לפתור זאת, במקום להטיל קרן אחת מכל נקודה על האובייקט לכיוון מקור האור כדי לראות אם הנקודה מוצלת או שלא, אנו "יורים" מספר רב קרניים לכיוון "רדיוס" מסביב למקור האור.

כדי לחשב את גוון הצל בכל נקודה על האובייקט, אנו מחשבים את הממוצע של כל הקרניים היוצאות מאותה הנקודה חזרה אל עבר מקור האור. וממוצע כל הקרניים הוא רמת ההצללה באותה הנקודה. שינוי זה גורם שבנקודות שקרובות יותר לקצה האובייקט, חלק מהקרניים לא יחתכו את האובייקט ויחזרו ישירות למקור האור מה שיוסיף קצת אור לצבע הנקודה. לאחר הוספת כל צבעי הקרניים נקבל צל שהולך ונהיה מואר יותר ויותר וזה נותן לנו את ההשפעה של הצללה דועכת כפי שרצינו.

**אופן מימוש השיפור:**

בפונקציה calcLocalEffects הוספנו קטע קוד שמחשב את ממוצע הגוון של אותה נקודה כפי שפרטנו לעיל:

תמונה שמכילה טקסט, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטיהוספנו גם פונקציית עזר לבניית הקרניים הרבות מהנקודה על האובייקט חזרה למקור האור:

**הדגמת השיפור:**

מבט מקרוב על צל בעייתי בסצנה:

אותה תמונה לאחר שיפור soft shadow:

תמונה שמכילה צבעוני, מג'נטה, ורוד, סיגלית

התיאור נוצר באופן אוטומטי

# שיפורי ביצועים

BVH - bounding volume hierarchy:

על מנת לחסוך את חישוב חיתוך הקרניים עם כל האובייקטים בסצנה (כלומר עבור כל קרן נצטרך לבדוק את כל הגיאומטריות), נבצע חסימה של כל אובייקט בסצנה בקופסה חוסמת (bounding box) שתכיל בצורה מלאה את האובייקט (למעט אובייקטים אינסופיים, שלא ניתן להגדיר עבורם קופסה חוסמת).

לאחר מכן נבצע בנייה אוטומטית של עץ היררכי בעזרת הפונקציה הבאה, buildTree:

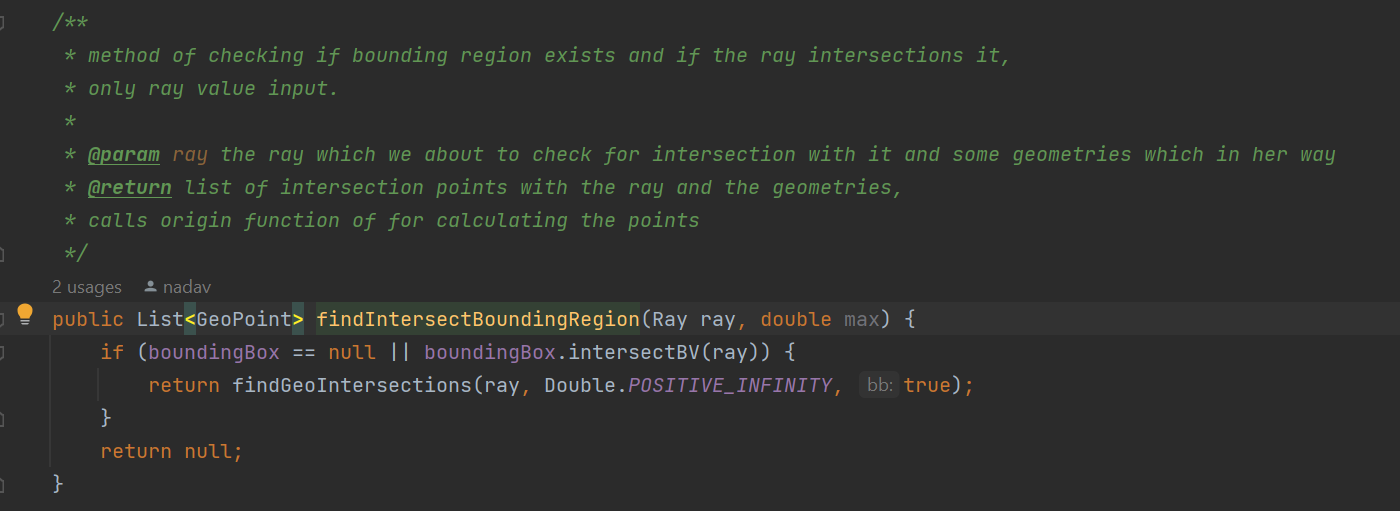


**הסבר על הפונקציה:**

1. שיטוח של כל הגיאומטריות (למקרה שיש גיאומטריות מורכבות, כמו קובייה/פירמידה/וכד').
2. כל עוד אין שורש עץ מוגדר (כלומר אין קופסה מכילה של הכל):
   1. מצא את שני המועמדים הטובים ביותר להכנסה לקופסה
   2. הכנס אותם לקופסה
   3. הכנס את הקופסה במקומם ל־containers.

בסוף הלולאה נקבל עץ בינארי בעל שורש בודד, שמייצג היררכיה של כל הגיאומטריות בסצנה.

לאחר יצירת העץ ההיררכי, ה־RayTracer שלנו ימשיך לעבוד כרגיל, אבל במקום לבדוק חיתוכים של הקרן עם כל גיאומטריה בעזרת הפונקציה האיטית findGeoIntersections שמשתמשת בחישובים מתמטיים מורכבים ו"כבדים" מבחינת זמן ביצוע, נשתמש בפונקציה findIntersectBoundingRegion, שהמימוש שלה כדלהלן:



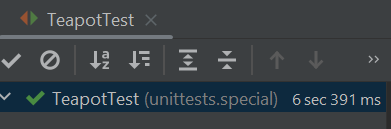
כלומר, נבדוק אם קיימת קופסה חוסמת וגם אם הקרן חותכת את הקופסא החוסמת (בצורה ה'קלה' מבחינה ביצועית). אם אחד משני התנאים לא מתקיים נחזיר את צבע רקע התמונה.

רק אם הקרן חותכת את הקופסא נבצע בדיקת חיתוך. במקרה שלא קיימת קופסא חוסמת הגענו לעלה בעץ ואין ברירה אלא לבדוק חיתוך עם גיאומטריה ממש.

**איכות שיפור הביצוע של BVH** (ללא Multi-Threading)

תמונה א' הקומקום (כפי שקיבלנו מוכן):

עם קופסאות חוסמות ועץ ההיררכיה:

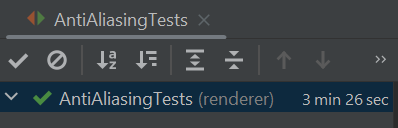
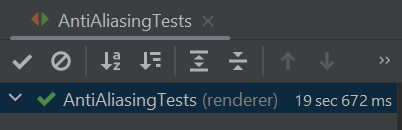
תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטיללא כל שיפורי BVH:

בתמונה זו **חל שיפור של** **פי 7.5** ( )בין התמונה לפני השיפור והתמונה אחרי השיפור!

תמונה ב' (חדר השינה):

עם קופסאות חוסמות ועץ ההיררכיה:

ללא כל שיפורי BVH:

בתמונה זו **חל שיפור של** **פי 11** ( )בין התמונה לפני השיפור והתמונה אחרי השיפור!

* נדגיש כי השיפור יבוא לידי ביטוי בצורה שונה בכל פעם, בהתאם למורכבות הסצנה, ובהתאם לכמות הגיאומטריות המוכלות בה ופיזורן במרחב.

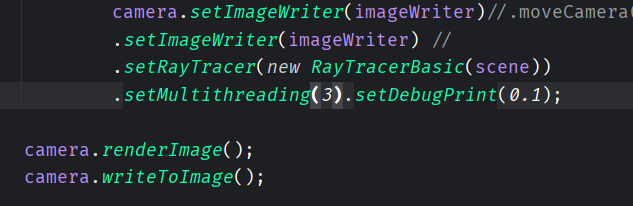
Multi-Threading:

עד עכשיו הרצנו את בניית התמונה באמצעות תהליך אחד בלבד.  
כדי לשפר ולייעל את מהירות עיבוד התמונה הוספנו multi-threading לפונקציית render Image   
אולם אז היינו חייבים לוודא שאין תהליך שדורס פיקסל של תהליך אחר ושעיבוד התמונה יעשה בצורה בטוחה. לשם כך הוספנו מחלקה חדשה: Pixel Manager שאחראי על קביעת הפיקסל הבא לעיבוד תוך שמירת ההפרדה בין התהליכים ווידוא שכל תהליך מבצע רק את מה שהוא נדרש לבצע

**לאחר הוספת השיפור השגנו שיפור ביצועים של לפחות פי 2.5**כשהשתמשנו ב-3 תהליכונים נצפה שיפור של פי 2.5 בערך בזמן ריצה עבור תמונות כבדות יחסית בהשוואה לרינדור ללא תהליכונים.

על מנת להפעיל את השיפור, יש להגדיר ב־RayTracer המוגדר בטסט:

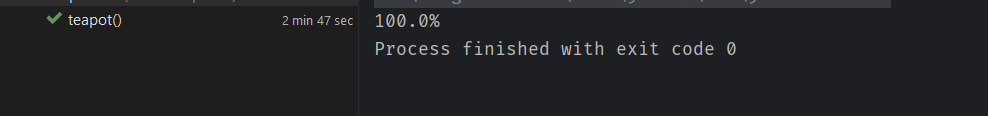
* הפעלת Multi-Threading עם הגדרת מספר התהליכונים שנרצה להקצות
* הפעלת הדפסת האחוזים
* הדגמה של הפעלת האפקט

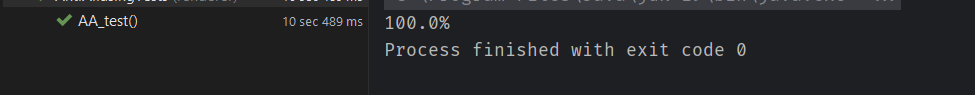


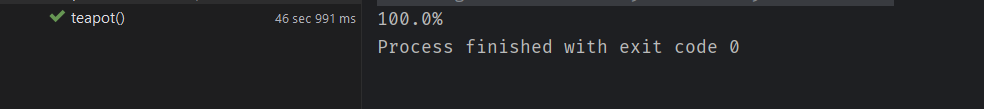
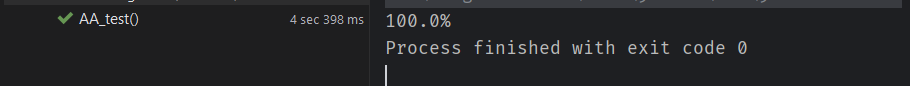
בתמונה המשופרת למעלה קבענו:

את מספר התהליכונים המוקצים לתיצוּג (רינדור) ל־3.

הפעלת הדפסת האחוזים.

זמן ריצה לפני השיפור (AA\_test הורץ בנוסף לשיפור bvh וteapot - הורץ ללא תוספת של BVH):



זמן ריצה לאחר השיפור (הוקצו 3 ליבות), ובנוסף הדפסת אחוזי התקדמות:

**בסך הכל שיפור של בין פי 2.5 (AA\_test) לפי 3.5 (teapot).**

# בונוסים:

* בדיקות + מימוש חיתוכי קרן ומצולע
* בדיקות + מימוש חיתוכי קרן וגליל (סופי)
* טרנספורמציית הזזה וסיבוב (של המצלמה)
* בניית סצנה מקובץ XML
* סיום שלב 6 תוך שבוע
* קבלת מקור תאורה ספוט עם אלומת אור צרה יותר
* בניית תמונה הכוללת גופים רבים (10 או יותר)
* הצגת התמונה מהבונוס הנ"ל מזוויות שונות וממרחקים שונים
* מימוש דרך 2 של פתרון בעיית ההצללה
* בונוסים שלב 8 (+ עשינו שני שיפורי תמונה)
* בונוסים שלב 9 – מיני פרוייקט 2

1. במקור (כפי שמופיע בgeogebra) מידות התמונה בתחום הZ מ-800-800, אולם בשל חשש מנקודה (0,0,0) התמונה עלתה ב800 וכעת תחום הZ מ0 עד 1600. [↑](#footnote-ref-2)