**שלב 7 - הוספת הצללה, שקיפות והשתקפות, השפעת שקיפות על הצללה (טיוטה)**

לפני שממשכים לשלב הזה יש לוודא שלא שכחתם ליצור **tag** ב-**commit** האחרון עבור השלב הקודם.

***נ.ב.*** *חובה להקפיד על תיעוד בפורמט* ***javadoc*** *לכל המחלקות והפעולות החדשות ולעבוד ע"פ הסדר כפי שנלמד בקורס!*

**חלק 0 – תיקון לשלב הקודם**

1. במחלקת **Render**:
   1. למחוק שדה של סצינה
   2. בסטר של סצינה - למחוק שורת עדכון שדה שנמחק לעיל
2. במקום סעיף 1ב לעיל – למחוק הפעלת הסטר של סצינה מהטסטים של שני השלבים הקודמים

**חלק 1 - הצללה**

1. במחלקת **RayTracerBasic**:
   1. נוסיף קבוע עבור גודל הזזת ראשית קרניים עבור קרני הצללה (ניתן להקטין את הערך שלו בהתאם לסדרי גודל הצורות בתמונה שלכם כך שההזזה לא תהיה ניכרת לעין בתמונה):

**private** **static** **final** **double** ***DELTA*** = 0.1;

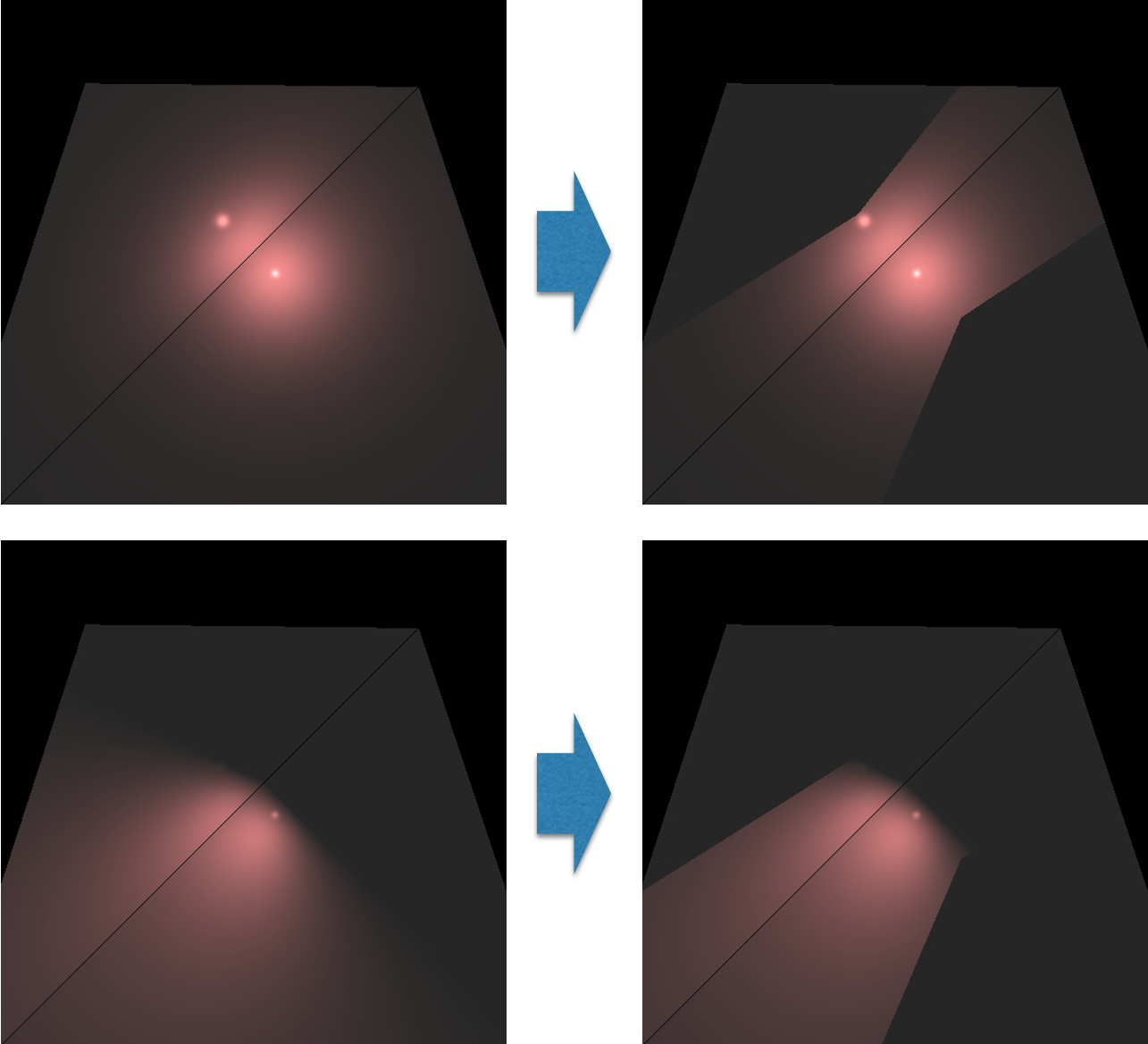
* 1. נוסיף פעולת בדיקת אי הצללה בין נקודה למקור האור שחתימתה (ניתן להוסיף פרמטרים אם אתם מוצאים שזה יועיל מבחינת הביצועים):

**private** **boolean** unshaded(Vector l, Vector n, GeoPoint gp)

**נ.ב.** ניתן להוסיף פרמטרים נוספים על מנת לשפר את הביצועים

* 1. כמובן לא נשכח להוסיף תיעוד בפורמט **JavaDoc** גם עבור הקבוע וגם עבור הפעולה הנ"ל
  2. נוסיף את בדיקת ההצללה בפונקציה **calcColor** בהתאם למה שנלמד בכיתה

1. נבצע את כל הטסטים **הישנים** ונוודא שהם עברו ושהתמונות לא השתבשו
2. נגלתה תקלה מכיוון בתמונת תאורה על פני זוג משולשים מופיע צל מהמשולש שנמצא מאחורי מקור התאורה:



על מנת לתקן את הבעיה נעשה **refactoring**:

* 1. נוסיף בפונקציה **unshaded** פרמטר מסוג **LightSource** ובזימון הפונקציה מעביר את מקור התאורה הנוכחי לפונקציה
  2. בממשק **LightSource** נוסיף חתימת פונקציה חדשה:

**double** getDistance(Point3D point)

* 1. נממש את הפעולה עבור שלושת מקורות התאורה המממשים את הממשק

**נ.ב.** במקור אור כיווני (**DirectionalLight**) נחזיר ערך **Double.POSITIVE\_INFINITY**

* 1. עכשיו "נסנן" את נקודות החיתוך באחד משני הדרכים:
     1. **דרך 1**: בפונקציה **unshaded,** אם רשימת החיתוכים שהתקבלה אינה ריקה – נעבור על הרשימה ואם נתקלנו בחיתוך שיותר קרוב לראש הקרן מאשר מרחק בין הנקודה לבין מקור האור – נחזיר **false**
     2. **דרך 2**: (לבונוס)
        1. נוסיף ונשנה את המתודות בממשק **Intersectable** כדלקמן (כמובן עליכם להשלים את כל התיעוד בפורמט **Javadoc** בהתאם):

**default** List<GeoPoint> findGeoIntersections(Ray ray) {

**return** findGeoIntersections(ray, Double.***POSITIVE\_INFINITY***);

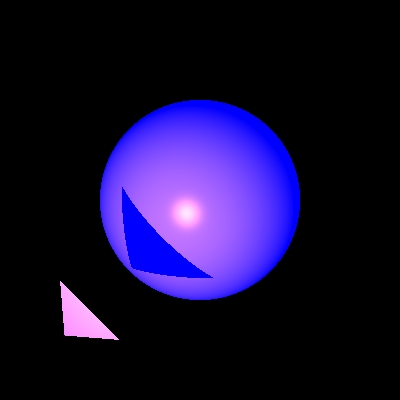
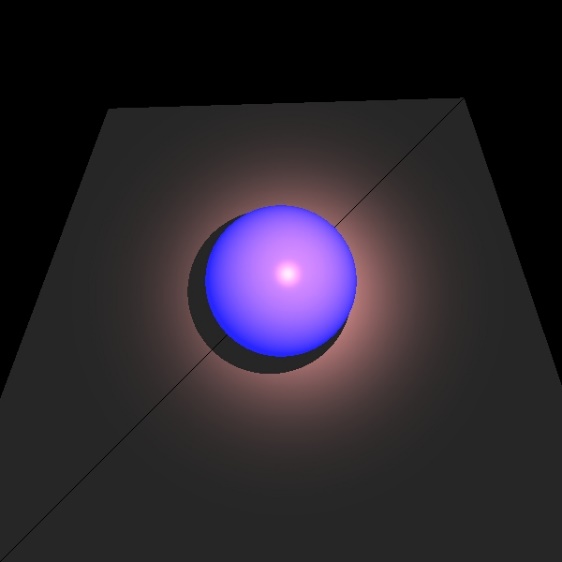
}  
List<GeoPoint> findGeoIntersections(Ray ray, **double** maxDistance);

* + - 1. בכל המחלקות של הגאומטריות שבהן ממומשת המתודה הישנה, נוסיף מימוש של המתודה החדשה כולל השינויים הנדרשים שמונעים הוספה אן בנייה של רשימה שכוללת נקודות שרחוקות מראש הקרן יותר מהמרחק המירבי **maxDistance**, למשל כך יכול להיראות התנאי:

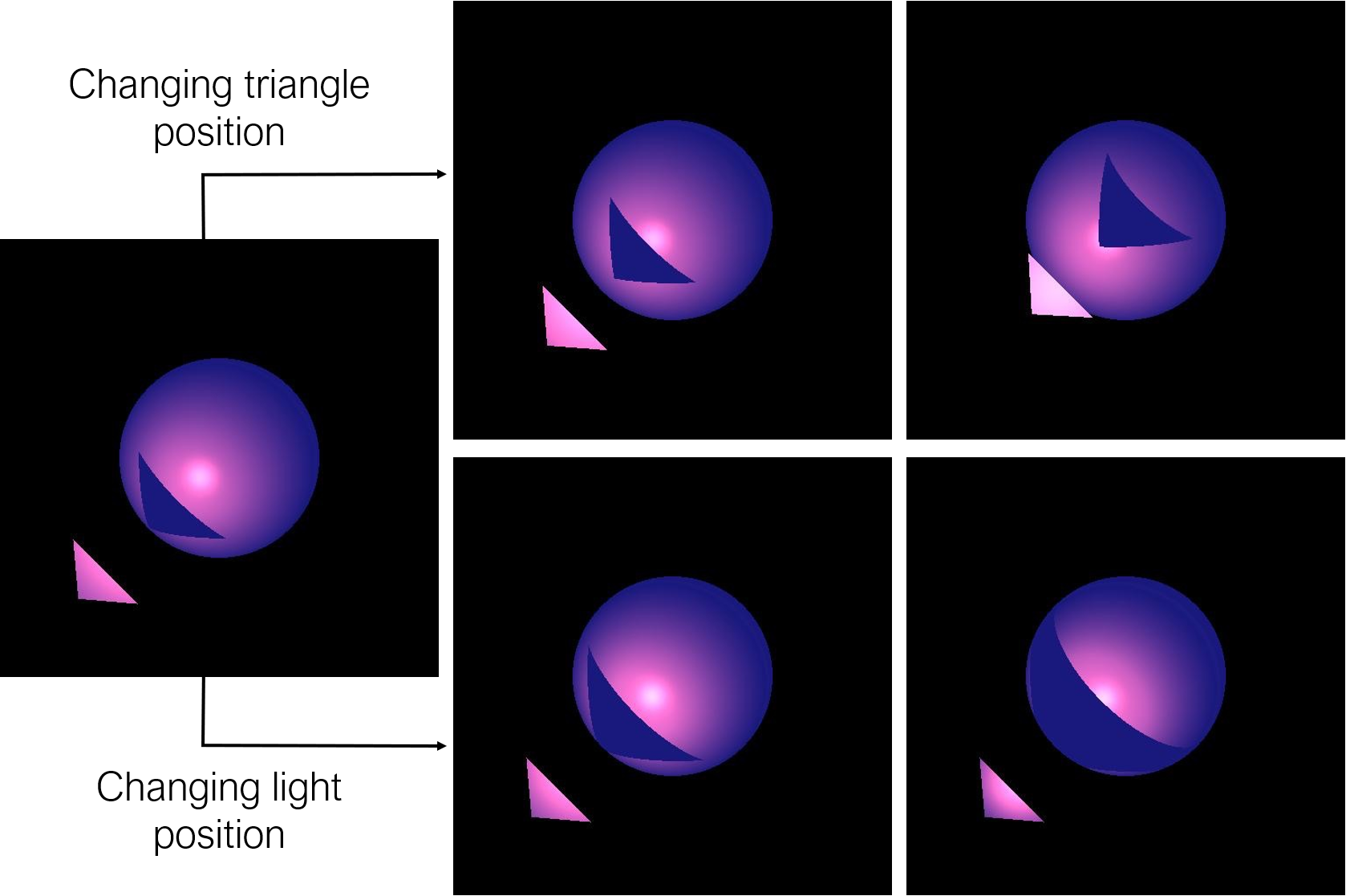
**alignZero(t – maxDistance) <= 0**

* + - 1. בפונקציה **unshaded** נוסיף לזימון של **findGeoIntersections** מרחק בין הנקודה למקור האור כארגומנט עבור הפרמטר החדש.
      2. נוסיף טסטים לבדיקת הפעולה החדשה לפי ההגיון שלכם. נריץ את **כל** הטסטים מחדש.
      3. נמחק את מימושי המתודה הישנה ממחלקות הגאומטריות. נריץ את **כל** הטסטים מחדש.
      4. אין צורך בשינויים נוספים
  1. נוודא שהבעיה תוקנה

1. נוריד מתיבת ההגשה של התרגיל את המודול **ShadowTests.java**, נשים אותו בחבילת **unittests** ונריץ אותו – נבדוק את התמונות:

** **

1. בטסט הזה ישנן 2 (שתי) פונקציות של בדיקת הצללה: משולש לפני ספרה וספרה מעל זוג משולשים. חובה עליכם להוסיף 4 פונקציות בדיקה (בהעתקה מהפונקציה הראשונה כנ"ל) ולעדכן את הנתונים בכל הפעולות שהוספתם על מנת לקבל תמונות עם הזזת המשולש והזזת מקור האור כפי שמופיע במצגת המעבדה:



1. נוסיף טאג 07.1 המייצג השלמת הוספת ההצללה

**חלק 2 – שקיפות והשתקפות**

1. נוסיף במחלקה **Material** שני שדות חדשים מסוג **double** בשם **kT** (מקדם שקיפות) ו-**kR** (מקדם השתקפות) בהרשאה ציבורית. יש להוסיף תיעוד בפורמט **javadoc** לשדות.
   1. שתי השדות יאותחלו ל-0.0
   2. נוסיף פעולות מעדכנות משרשרות (סטרים) בהתאם לתבנית **Builder** עבור השדות החדשים (עם תיעוד מתאים בפורמט **javadoc**)
2. במחלקת **RayTracerBasic**:
   1. נוסיף שני קבועים עבור תנאי עצירה ברקורסיה של שקיפויות\התשקפויות (ניתן לשנות את הערכים ע"פ הערכתכם והבנתכם שתצטרכו להסביר בהצגת השלב כמובן):

**private** **static** **final** **int** ***MAX\_CALC\_COLOR\_LEVEL*** = 10;

**private** **static** **final** **double** ***MIN\_CALC\_COLOR\_K*** = 0.001;

* 1. נוסיף פעולות חישוב קרן השתקפות וקרן שקיפות (כמובן עם תיעוד בפורמט **javadoc**) – בקרניים האלה תוזז נקודת תחילת הקרן על ישר הנורמל לגאומטריה בכיוון הקרן החדשה – בדומה למה שעשינו בבניית קרני הצללה
  2. מכיון שעכשיו נצטרך לחשב את החיתוך הקרוב ביותר ממקום אחד – נעשה **refactoring** ע"י בניית פעולה שתחשב את החיתוכים וגם תחשב את החיתוך הקרוב לראשית הקרן עם החתימה הבאה (כמובן עם תיעוד בפורמט **javadoc**), הפעולה תחזיר **null** אם לא היו נקודות חיתוך לקרן:

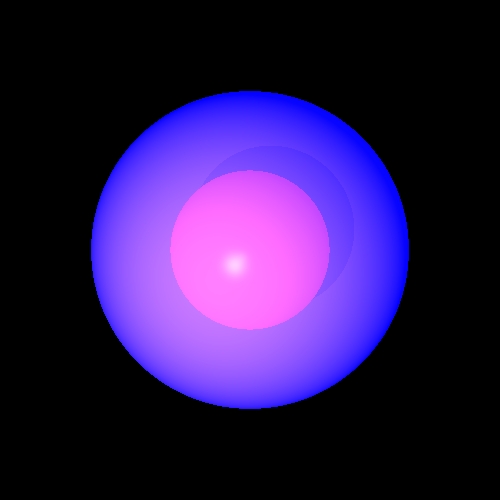
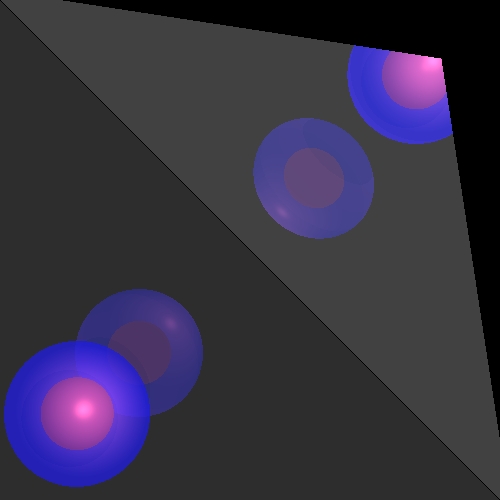
**private** GeoPoint findClosestIntersection(Ray ray)

* 1. נעשה עדכון בקוד של פעולת הרינדור כך שתשתמש בפעולה הזו החדשה
  2. נוסיף פעולה חדשה **calcColor** עם החתימה הבאה (כמובן עם תיעוד בפורמט **javadoc** – התיעוד יהיה מהפעולה המקורית עם עדכונים בהתאם) – הפעולה תזמן פעולת עזר כפי שנלמד בכיתה (ראו בסעיף הבא):

**private** Color calcColor(GeoPoint gp, Ray ray)

* 1. נשנה את הפעולה המקורית **calcColor** כך שתהפוך לפעולת רקורסיבית בהתאם למה שנלמד בכיתה (שימו לב שהוספת רכיב האורה הסביבתית יעבור לפעולה "הראשית" הנ"ל) – עם פרמטרים נוספים ועם תמיכה בשקיפות והשתקפות, כפי שנלמד בכיתה, כולל פעולות חדשות נוספות (עם תיעוד בפורמט **javadoc**)
  2. נעדכן את הפונקציה **unshaded** כך שרק אוביקטים ללא מקדם שקיפות (זאת אומרת מקדם שקיפות = 0) יגרמו להצללה (בחלק הזה) כפי שנלמד בכיתה
  3. נבצע את כל הטסטים הישנים ונוודא שהם עברו ושהתמונות לא השתבשו

1. נוסיף לחבילה **unittests** מודול **ReflectionRefractionTests.java** מתיבת ההגשה – המודול כולל את שני הטסטים שהוצגו בכיתה (שני כדורים מקוננים לבד, שני כדורים מקוננים לפני שני משולשים של מראה, בינתיים נתעלם מהתמונה השלישית שנוצרת במודול הזה):

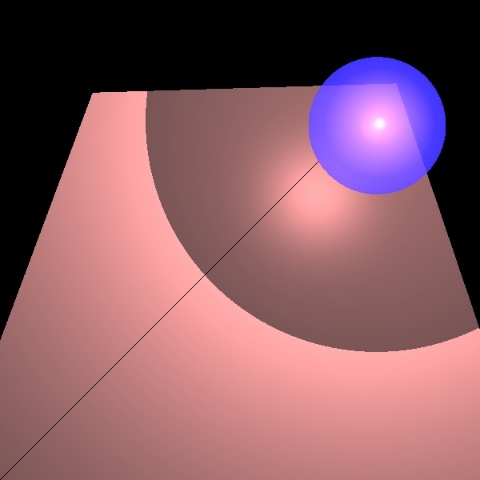
1. נשים לב על כך שבניית קרן עם הזזת נקודה – זו פעולה שחוזרת על עצמה כבר שלוש פעמים, לכן כדאי להוריד אותה לפונקציה נפרדת. מיקום הפונקציה ע"פ **RDD** הוא מחלקת הקרן. מכיוון שמדובר בבניית קרן חדשה (ע"פ נקודה מקורית, כיוון הקרן, וקטור הנורמל (שעל הישר שהוא מגדיר יש להזיז את הנקודה של ראש הקרן)) – נוסיף את הפונקציה כבנאי נוסף של קרן ונעדכן בכל המקומות הרלוונטיים את הקוד כך שישתמש בבנאי החדש של הקרן. כמובן הקבוע **DELTA** שהגדרנו קודם יועבר למחלקת **Ray**. נריץ את כל הטסטים ונוודא את תקינות תוצאותיהם.

**חלק 3 – שקיפות והצללה חלקית**

נרצה לקבל **הצללה חלקית** במקרה שהגוף או הגופים שחסומים את מקור האור מהנקודה הינם בעלי שקיפות ברמה כזו או אחרת. על מנת להשיג מטרה זו:

1. נעתיק את הפונקציה **unshaded** ובעתק נשנה את השם ואת סוג הערך המוחזר כדלקמן:

**private** **doulbe** transparency(LightSource ls, Vector l, Vector n, GeoPoint geoPoint)

1. נוסיף לפונקציה תיעוד בפורמט **javadoc** ונשנה את המימוש של הפעולה כפי שנלמד בכיתה ומפיע במצגת המעבדה
2. בפונקציה **calcColor** נחליף את הקריאה ל-**unshaded** ונשתמש בפונקציה החדש בהתאם, כפי שנלמד בכיתה ומופיע במצגת המעבדה
3. שימו לב: הקוד במצגת הינו פסיודו-קוד ולא חייב להיות תקין ב-100% מבחינת תחביר java
4. נריץ את **כל** הטסטים מחדש ונעבור על כל התמונות – נוודא שהכל תקין והתמונה האחרונה החדשה נראית כמו:
5. נבנה תמונה משלנו (חובה!) שמשלבת 3-4 גופים שונים שמאפשרת להראות את כל האפקטים החדשים ביחד
6. נ.ב. אל תיצרו אובייקטים סגורים (למשל כדור) בעלי שקיפות והשתקפות בו זמנית!!! אחרת בניית התמונה תיקח לכם זמן רב בגלל הקרניים שנלכדות ומטיילות בתוך הכדור

**חלק לבונוסים**

ניתן לקבל בונוס של עד 3 נק' ע"י (1 נק' עבור כל אחד מהסעיפים):

* בניית תמונה הכוללת גופים רבים (10 או יותר) גופי **שונים** (נכל הסוגים שמימשתם!) תוך הדגמת כל התכונות ואפקטים שבוצעו עד כה, כולל השלב הזה – לא תתקבל תמונה עם אותו אחד או אותם שני אובייקטים שמוקמו בצורה רנדומלית – התמונה חייבת להיות מרשימה.
* הצגת התמונה מהבונוס הנ"ל מזוויות שונות וממרחקים שונים (ע"י הזזת הקמרה). לא תתקלנה תמונות של סתם קירוב הקמרה על אותו ציר כמו במיקום אחד. יש להשתמש בבונוס משלב 4 (השלימוהו אם עוד לא עשיתם אותו)
* מימוש **דרך 2:** של פתרון בעיית ההצללה

בהצלחה!