מיני פרויקט שלב 1

**Primitives**

1. Coordinate - יחידה על ציר המספרים.
2. Point2D - נקודה במישור.
3. Point3D - נקודה במרחב.
4. Vector - ישר היוצא מראשית הצירים מקבל את כיוונו ע"י נקודה.
5. Ray - וקטור שלא עובר בראשית הצירים והוא מוגדר ע"י נקודה וכיוון.
6. Material - מחלקה שאחראית על שימוש בחומר.

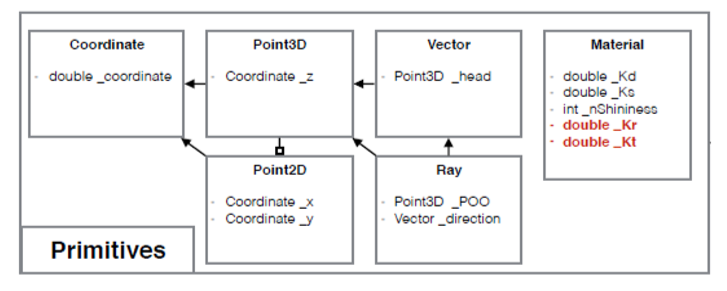
Geometries

1. Geometry- מחלקה אבסטרקטית עבור גוף גיאומטרי מסוים.
2. RadialGeometry - מחלקה המייצגת ורה גיאומטרית מעגלית.
3. FlatGeometry - ממשק המייצג צורות שטוחות.

מטרה שלב 1: מטרת המיני פרויקט בשלב א' היא לבנות מהבסיס מיצג גרפי תלת-ממדי, נעשה זאת ע"י החבילה primitives הכוללת מחלקות בסיסיות באלגברה ובגיאומטריה.

Primitives

מבנה החבילה:



*החבילה מכילה פונקציות שעושות פעולות בסיסיות בקואורדינאטות ובווקטורים במישור ובמרחב.*

החבילה כוללת את המחלקות הבאות:

1. Coordinate- יחידה על ציר המספרים. המחלקה מכילה:

משתנה coordinate מסוג double. בנאים ופונקציות טריוויאליות (Getters and Setters)

**פונקציות:**

א.פונקציית Boolean equals המשווה בין וקטורים ובעת הצורך משתמשת בפונ'

usubtract שמצויה במחלקת utils לזיהוי ערך שקרוב לאפס כאפס.

ב. פונ' toString שמדפיס את הערך המוחזר בעשרוני.

ג. פונ' compareTo-שמחזירה 0 אם שווים 1 אם גדול ו1- אם קטן ממנו.

ד. פונ' subtract-חיסור קואורדינטה מהנוכחית.

ה. פונ' add- מקבלת קואורדינטה ומשנה את הנוכחית ע"י תוצאת החיבור ביניהם.

ו. פונ' scale- מקבלת מספר ומשנה את הנוכחית ע"י תוצאת הכפל ביניהם.

ז. פונ' multiply- מקבלת קואורדינטה ומשנה את הנוכחית ע"י תוצאת הכפל ביניהם.

1. Point2D- נקודה במישור המכילה שתי קואורדינטות, מסוג coordinate.

**פונקציות:**

א.בנאים ופונקציות טריויאליות.

ב. compareTo-משווה בין נקודות, מחזירה 0 אם שווים 1 אם גדול ו1- אם קטן ממנו.

ג. toString-מחזירה את ערך הנקודה במישור בפורמט הדפסה.

ד -distanceBetween2DPoints.מקבלת נקודה ומחזירה את המרחק ביניהן.

1. Point3D - נקודה במרחב המכילה שלוש קואורדינטות, מסוג coordinate.

תיאור-המחלקה point3Dיורשת מpoint2D ומוסיפה ערך של קואורדינטה שלישית.

ישנם ארבעה בנאים:

א.ברירת מחדל – מציב נקודה בתחילת הצירים בערך (0,0,0).

ב. בנאי שמקבל 3 ערכים מסוג coordinate.

ג.בנאי שמקבל שלושה ערכים מטיפוס double.

ד.בנאי העתקה

**פונקציות:**

א. compareTo-משווה בין נקודות, מחזירה 0 אם שווים 1 אם גדול ו1- אם קטן ממנו.

ב. toString-מחזירה את ערך הנקודה במרחב בפורמט הדפסה.

ג.divide-מחלק את הוקטור במספר עשרוני.

ד.add-הוספה של וקטור.

ה.subtract-חיסור וקטור.

ו.distance- מקבלת נקודה ומחזירה את המרחק ביניהן.

1. Vector - ישר היוצא מראשית הצירים מקבל את כיוונו ע"י נקודה. מכיל משתנה \_head מסוג point3D.

5 בנאים:

א. ברירת מחדל – מגדיר את ראש הווקטור בתחילת הצירים בערך (0,0,0).

ב. בנאי שמקבל ערך מטיפוס Point3D מכניס את ערכו למשתנה המקומי.

ג. בנאי שמקבל שני פרמטרים מסוג Point3D ומכניס את ערך החיסור בניהם למשתנה המקומי של המחלקה. שתי הנקודות מייצגות וקטור, החיסור בניהם מביא את אותו ווקטור עם אותו אורך וכיוון, רק שהוא מתחיל מראשית הצירים ויותר קל לייצג אותו.

ד. בנאי העתקה

**פונקציות:**

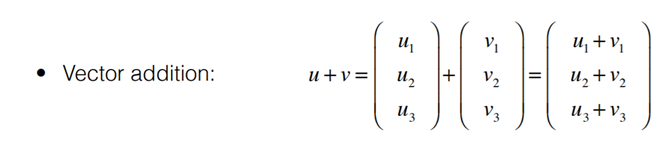
א. פונקציות טריוויאליות.

ב. compareTo- ההשוואה בפונקציה זו מתבססת על ההשוואה ב Point3D

לפי אורך הווקטור.

ג. toString-משתמשת בtoString של point3D.

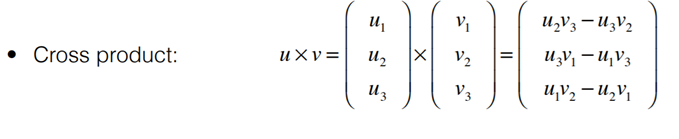
ד. add- הוספת וקטור לווקטור הנוכחי.



ה. subtract -חיסור וקטור מהווקטור מהנוכחי.

ו. scale -הכפלה של הווקטור בקבוע.

ז. CrossProduct -מבצע כפל וקטורי ומחזיר את התוצאה.



ח. length-מחזיר אורך וקטור.

ט. normalize-נרמול וקטור.

י. dotProduct-מחזיר תוצאה סקאלרית בכפל בין שני וקטורים.

1. Ray - וקטור שלא עובר בראשית הצירים והוא מוגדר ע"י נקודה וכיוון.

מכיל Point3D \_POO ו Vector \_direction-

3 בנאים:

1. ברירת מחדל – מגדיר את הנקודה והכיוון על (0,0,0) , מיקום בראשית הצירים.
2. בנאי שמקבל – נקודת מקור מטיפוס Point3D, וכיוון מטיפוס Vector.  
   מכניס אותם למשתנים המקומיים.
3. בנאי העתקה

**פונקציות:**

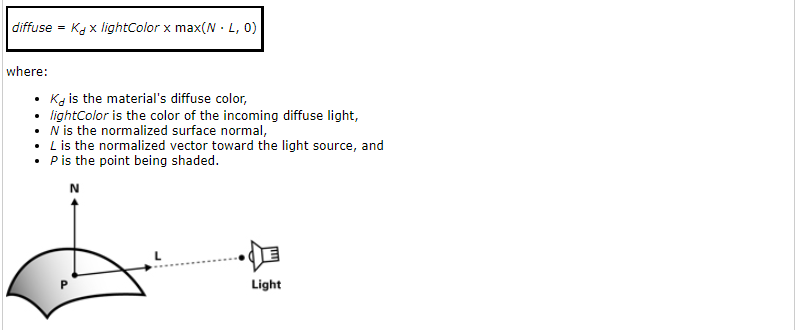
א. פונקציות טריויאליות.

ב. compareTo- משווה בין קרנות, ומחזירה 0 אם שווים 1 אם גדול ו1- אם קטן ממנו.

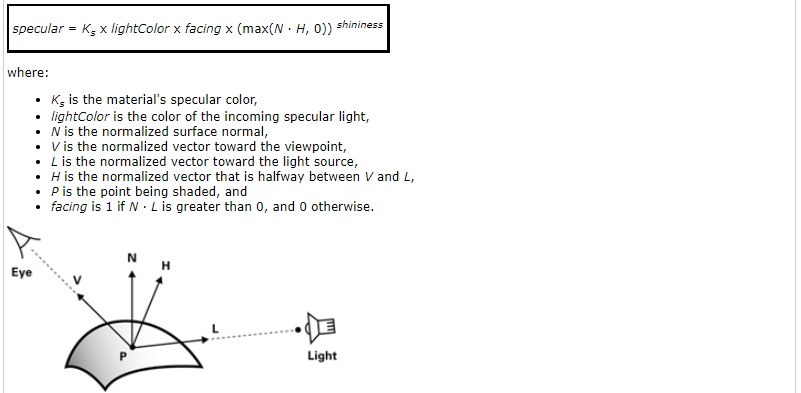
ג. toString - מחזירה את ערך הנקודה והכיוון בפורמט הדפסה.

6) Material - מחלקה שאחראית על שימוש בחומר. מכילה משתנים כדלהלן:

1. double \_Kd-מקדם פיזור



1. double \_Ks- מקדם הנחתה ניטרלי.



1. double \_Kr -מקדם השתקפות (מראה אחת).
2. double \_Kt מקדם השתקפות (עבור שקף)
3. double \_n - מקדם השבירה.

בנוסף ישנם בנאים ופונקציות טריוויאליות.

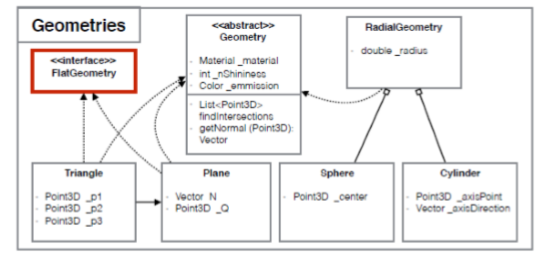
מיני פרויקט שלב 2

Geometries

בעזרת חבילת הפרימיטיביים שהגדרנו נוכל עכשיו להגדיר אוסף של מחלקות בעזרתם נוכל לתאר צורות וגופים הנדסיים.

נגדיר חבילה בשם **geometries** שיש לה אוסף מחלקות המתארות צורות וגופים גיאומטריים, ע"מ לבנות סצנה גרפית מגופים מגוונים נבנה את החבילה בצורה היררכית.

מבנה:



מחלקות חבילת Geometries

מחלקות אבסטרקטיות:

1. –Geometry מחלקה אבסטרקטית עבור גוף גיאומטרי מסוים (מחלקות שונות של הגופים השונים ירשו ממנה). המחלקה תכיל בנאי ברירת מחדל והעתקה, 3 משתנים של סוג החומר, רמת ההברקה והצבע. כמו כן סטרים וגטרים.

**פונקציות:**

א. פונקציה אבסטרקטית **getNormal** המקבלת פרמטר יחיד של נקודה ומחזירה וקטור אנך לנקודה זו.

ב. פונקציה אבסטרקטית **findIntersection** שמקבלת משתנה מסוג Ray ומחזירה רשימה של נקודות החיתוך עם הצורה הגיאומטרית.

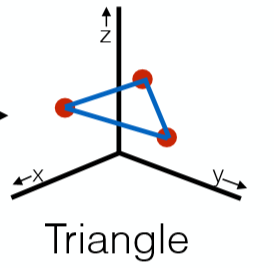
1. RadialGeometry- מחלקה אבסטרקטית היורשת מgeometry וכוללת בנאים (בנאי ברירת מחדל שמגדיר את הרדיוס באורך 0 ובנאי שמקבל משתנה עשרוני) שדה רדיוס וגטר עבורו. תפקידה לייצג צורה גיאומטרית מעגלית.
2. מחלקת :FlatGeometry  
   ממשק לצורות גאומטריות שאינן מעגליות, שטוחות.  
   אין לממשק פונקציות, אלא משתמשים בו במחלקת Render בפונקציה Occluded, על מנת למנוע שגיאות ביצירת צורות גאומטריות שטוחות בתמונה.
3. Intersectable interface:

מכיוון שאנו צריכים למצוא נקודות חיתוך עם כל גיאומטריה, נגדיר ממשק(interface) שבו יש פונקציה וירטואלית טהורה בשם FindIntersections (מציאת נקודות חיתוך), שמחזירה רשימה של נקודות תלת-ממדיות. ברשימה נמצאות כל נקודות החיתוך של קרן מסוימת עם הגיאומטריה שמממשת את אותו ממשק.

מחלקות הגופים המגוונים:

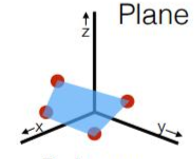
המחלקות שנמנה יורשות מהמחלקות האבסטרקטיות. **צורות שטוחות**:

1. משולשtriangle/-3 נקודות במרחב. מממשת את הממשק השטוח. ישנם 3 בנאים (בנאי ברירת מחדל, ההעתקה ובנאי המקבל שלוש נקודות) גטרים וסטרים, **findIntersection** ו-GetNormal.

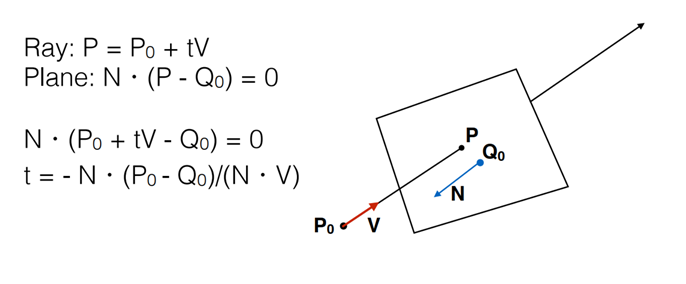


1. מישורplane/- נקודה במרחב ווקטור אנך. ישנם שני משתנים:vector המייצג נורמל למישור ונקודה במישור. בנאי ב"מ, העתקה והשמה.גטרים וסטרים.

פונ' **findintersection** המייצגת את נקודת החיתוך של הקרן עם המישור.



**הנוסחה ל- findIntersection**:

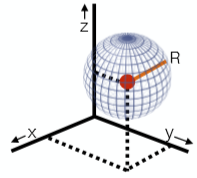


**צורות תלת ממדיות:**

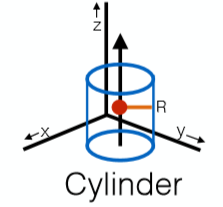
1. כדורsphere/ - נקודה עם רדיוס. המחלקה יורשת מ **RadialGeometry**- יש משתנה אחד המייצג את מרכז המעגל, ישנם שלושה בנאים גטרים סטרים. פונקציות: GetNormal-המחזירה וקטור נורמל לנקודה בכדור שהתקבלה כפרמטר.

**FindIntersection**-המחזירה רשימה של נקודות חיתוך עם הכדור.

**getPoint**– פונקציית עזר לחישוב נקודות החיתוך עם הכדור.

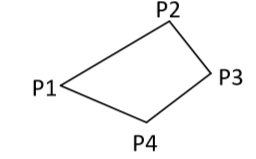


1. גלילcylinder/ - קרן גובה ורדיוס. המחלקה יורשת מ **RadialGeometry-** ישנם שני משתנים אחד המייצג וקטור של כיוון הגליל ושני המייצג נקודה בגליל.ישנם סטרים, גטרים ושלושה בנאים. כמו כן פונקציות GetNormal ו- **Findintersection.**



1. Quadrangle - מחלקה היורשת ממחלקת geometry ומממשת את ממשק flatGeomtry מרובע המבוסס על שני משולשים. בנאי המקבל ארבעה משתני נקודות.

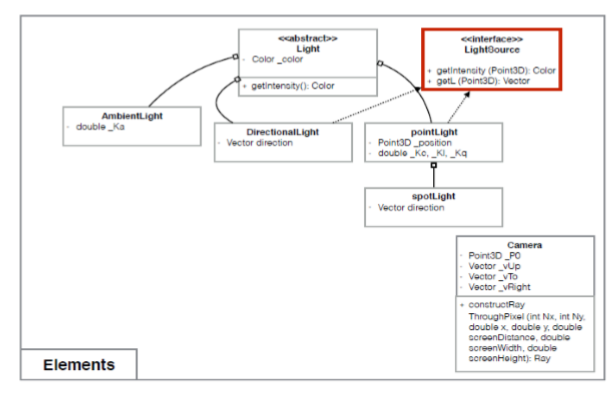
ישנם סטרים, גטרים ובנאי אחד המקבל 4 נקודות. כמו כן פונקציות GetNormal ו- **Findintersection.**

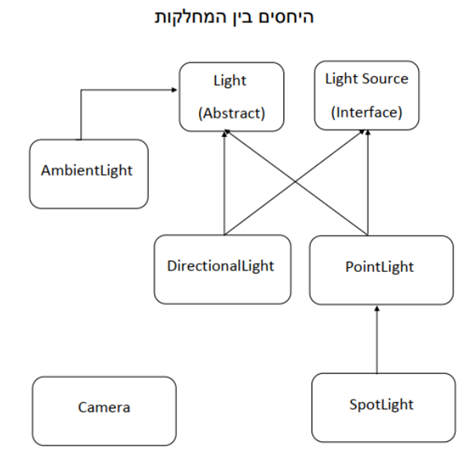


Elements

אחרי שהגדרנו את המודל התלת-ממדי שאותו רוצים לייצג ,בשלב זה של הפרויקט נרצה להגדיר את המצלמה שהיא נקודת המבט שלנו על המודל שלנו ,על הצורות הגיאומטריות ומקורות האור שמשתמשים בהם בסצנה.

מבנה:





ישנן שש מחלקות בחבילת **elements**:

1. **Camera**

3 בנאים:

א.בנאי ברירת מחדל – מגדיר את עין המצלמה ב (0,0,0), הכיוונים:

מעלה – (0,1,0)

מול – (1-,0,0)

ימינה – תוצאת מכפלה סקלרית בין 2 הכיוונים השניים.

ב.בנאי שמקבל 3 ערכים – מסוגים Point3D, Vector, Vector -מכניס את הערכים למשתנים המקומיים.

ג.בנאי העתקה.

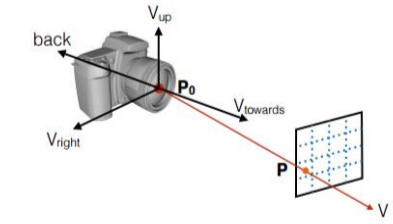
**משתנים:**

א. Point3D -נקודה תלת ממדית שהיא מרכז המצלמה המשקיף על המודל מראשית הצירים.

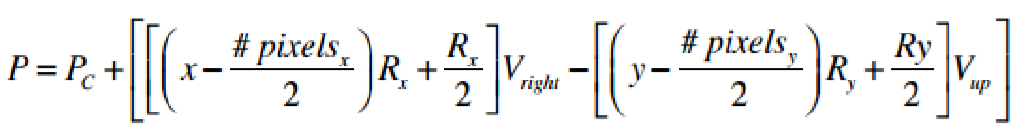
ב. ממרכז המצלמה יוצאים שלושה וקטורים שהם הצירים של המערכת- וקטורי הכיוון:

* Vup - וקטור בכיוון החיובי של המערכת כלפי מעלה, ההמשך שלו בכיוון ההפוך הוא הווקטור שמצביע כלפי מטה.
* Vright - וקטור לכיוון הצד הימני של המערכת, ההמשך שלו בכיוון ההפוך הוא הווקטור שמציין את הצד השמאלי.
* Vtoward - וקטור שכיוונו לקראת מישור הצפייה, ההמשך שלו בכיוון ההפוך הוא הווקטור שפונה לכיוון ההפוך ממישור הצפייה.

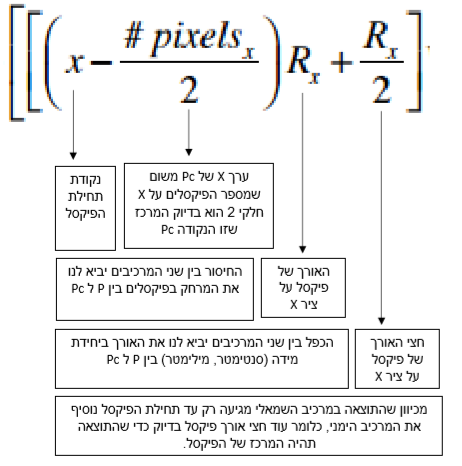
המחשה:



**פונקציית** . constructRayThroughPixel- פונקציה שהיא בונה קרן דרך פיקסל, כלומר מישור הצפייה שלנו בנוי ומחולק לפי פיקסלים כאשר השטח הוא H\*W. אנו נרצה לשלוח וקטור ממרכז ההקרנה בכיוון מסוים לעבר מישור הצפייה כאשר הוא עובר דרך פיקסל מסוים.

הנוסחה היא:   

הסבר:



ד. פונקציות גטרים סטרים toString שמדפיסה את נתוני המצלמה.

1. **אור/light**

מחקה אבסטרקטית המייצגת אור.

למחלקה יש משתנה אחד מטיפוס color כמו כן ישנם שני בנאים: א.בנאי ברירת מחדל שמגדיר צבע שחור. ב. בנאי שמקבל צבע ומבצע השמה למשתנה הצבע של המחלקה.

פונקציית **getIntensity**- מחזירה ערך מטיפוס color, שמייצג את האור ע"פ צבעו. וגטרים וסטרים.

1. **Interface: LightSource**

ממשק שמייצג מקורות אור חיצוניים.

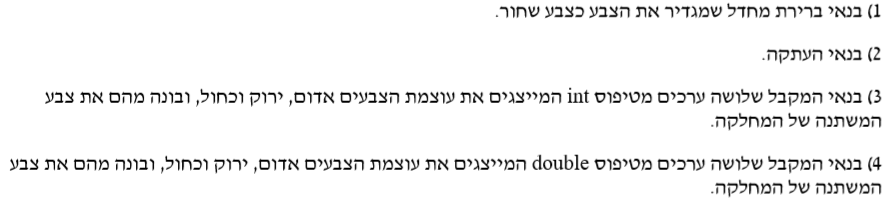
**פונקציות:**

**getIntensity**- פונקצייה שמקבלת טיפוס מסוג point 3D ומחזירה ערך מטיפוס color-שמייצג את עוצמת האור.

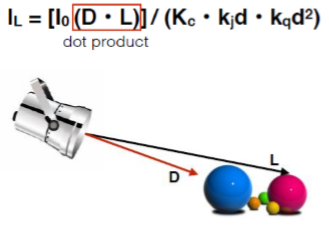
**getL**- פונקצייה שמקבלת טיפוס מסוג point 3D ומחזירה וקטור ממקור האור לנקודה זו.

1. **AmbientLight**מחלקה שמייצגת אור סביבתי, יורשת מlight-. ישנו משתנה שמייצג קבוע שמחליש את האור.

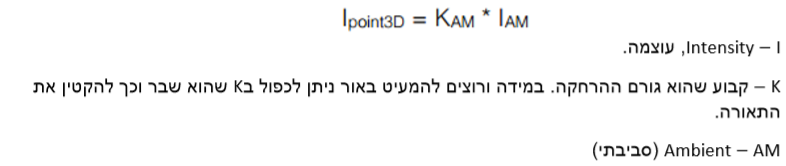
למחלקה ישנם ארבעה בנאים:



פונקציות גטרים וסטרים, ומימוש לממשק equal.

**getIntensity** - פונקצייה שמקבלת טיפוס מסוג point 3D ומחזירה ערך מטיפוס color - שמייצג את עוצמת האור ע"י הכפלה. הנוסחה: 

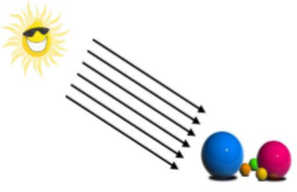
פונקציית **toString** ע"מ לבצע הדפסת האובייקט.

תאורה סביבתית - מקור האור הסביבתי מייצג מקור אור בעל עוצמת אור גבוהה ומשפיע על כל האובייקטים בסצנה שווה. ישנה נוסחה: 

הסבר-הנוסחה אומרת שהעוצמה של התאורה בנקודה מסוימת () שווה לקבוע ההרחקה של התאורה הסביבתית כפול העוצמה של התאורה הסביבתית.

לאחר הביצוע, כל האובייקטים בסצנה מוארים בצבע ובעוצמה שצוינו, הסוג הפשוט ביותר של תאורה מיושמת, ומודלים- ללא תלות בכמה הם דלילים בפיזורם או כמות הפעמים שהם משתקפים- יוצרים אפקט אחיד.

1. **directionLight** - מחלקה היורשת ממחלקת Light ומממשת את ממשק LightSource. תפקידה לייצג אור עם כיוון, כמו קרני שמש, ע"י מתנה אחד מטיפוס וקטור. כמו כן יש בנאי המקבל טיפוס color ווקטור כיוון, וגטרים וסטרים.

המחשה: 

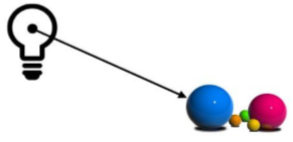
1. **pointLight** - המחלקה מייצגת נקודת הפקת אור, כמו מנורה. המחלקה יורשת ממחלקת Light ומממשת את ממשק LightSource.

למחלקה יש ארבעה משתנים, אחד מטיפוס point3D שמייצג את מיקום האור, ושלושה טיפוסים של גורמי הנחתת האור.

למחלקה יש בנאי אחד המקבל חמישה ערכים, צבע, מיקום ושלושה גורמי הנחתת האור. יש שימוש בפונקציית **getIntensity**לייצוג עוצמת האור. **נוסחא**:



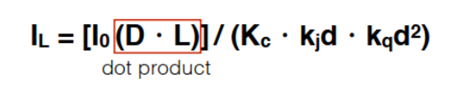
פונקציית**getL** – מקבלת משתנה מסוג point3D ומחזירה וקטור שמייצג את הוקטור מהאור לנקודה הזו.



1. **Spotlight** - המחלקה מייצגת נקודה שמייצרת אור עם כיוון. המחלקה יורשת מהמחלקה pointLight. יש למחלקה משתנה מטיפוס vector. כמו כן יש למחלקה בנאי אחד המקבל שישה ערכים, צבע מטיפוס color, מיקום מטיפוס point3D, כיוון מטיפוס וקטור ו-3 גורמי הנחתה.

פונקציית **getIntensity** - המייצגת את עוצמת האור.

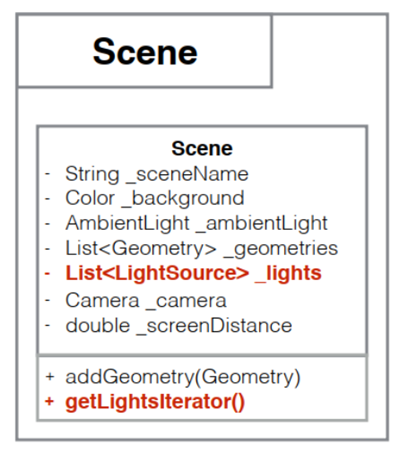
נוסחא:



Scene

חבילת scene מייצגת את מכלול הסצנה על ידי ריכוז הצורות הגיאומטריות בתמונה, הרקע והתאורה.

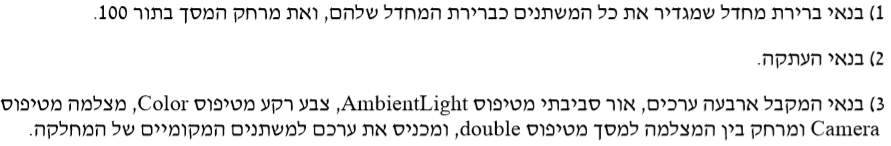
מבנה:



**מחלקת** **scene**

המחלקה מייצגת סצנה. למחלקה יש שבעה משתנים, אחד מטיפוס צבע המייצג את צבע הרקע של הסצנה, אחד מסוג ambientlight המייצג את התאורה הסביבתית של הסצנה. אחד מטיפוס LIST<GEOMETRY> שמייצג את רשימת הצורות הגיאומטריות,אחד מטיפוס camera, אחד המייצג את המרחק מהמצלמה למסך, אחד מטיפוס LIST<LightSource> המייצג את רשימת מקורות האור ואחרון מטיפוס string שהוא השם של הסצנה.

המחלקה עם שלושה בנאים:



**פונקציות**:

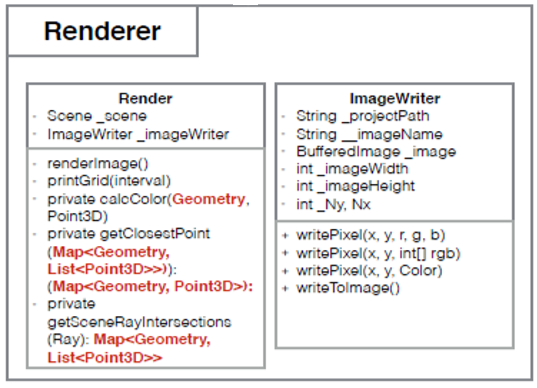
1. גטרים וסטרים.
2. **addGeometry**- מקבלת פרמטר גיאומטרי ומכניסה אותו לרשימת הצורות הגיאומטריות.
3. **addLight-**מקבל משתנה מטיפוס lightSource ומכניס אותו לרשימת מקורות האור של הסצנה.
4. **getGeometriesIterator**- מחזירה איטרטור לרשימת הצורות הגיאומטריות של הסצנה.
5. **getLightIterator**- יוצר איטרטור עוקב מטיפוס Iterator<LightSource> לרשימת מקורות האור של הסצנה.

מימוש לממשק equal ו-**toString**.

Renderer

חבילה שמייצגת את מחולל התמונה, המחשבת את הצבע שמתאים ומוציאה תמונה מתאימה.

מבנה:



1. **מחלקת** **ImageWriter**

מחלקה שמייצגת את מדפיס התמונה. ישנם שבעה משתנים: ארבעה המייצגים את גודל התמונה ומספר הפיקסלים שבה, קבוע מטיפוס string המייצג את מיקום הקובץ, אחד המייצג את התמונה ואחד המייצג את שם התמונה.

למחלקה יש שני בנאים :

* בנאי המקבל חמישה ערכים, ארבעה המייצגים את גודל הסמך ומספר הפיקסלים ואחד המייצג את שם התמונה.
* 2.בנאי העתקה.

**פונקציות:**

**WriteToImage** - מייצרת את התמונה לתוך קובץ מסוג jpg.

**WriteToPixel** - מציירת בפיקסל הנתון את הצבע שהיא קיבלה.

1. **מחלקת RENDER**

זו המחלקה המרכזית בפרויקט. מחלקה זו מייצרת לנו את התמונה ע"י קבלת הנתונים ממחלקת הסצנה ועיבודם עד הצגת התמונה בשלמותה, הצגת התמונה כוללת פרטים רבים כמו הצורות הגיאומטריות הקיימות בסצנה, המרחקים מכל צורה וצורה, מציאת המרחק הקטן ביותר, הצבע לכל צורה, האורות המשפיעים ביחס לצבע ועוד.

במחלקה יש שלושה משתנים: משתנה המייצג את הסצנה של התמונה, אחד מסוג ImageWriter שמייצג את המדפיס של התמונה וקבוע מסוג אינטננג'ר שמייצג את רמת הרקורסיה.

ישנו בנאי אחד המקבל שני ערכים ומכניס את ערכם למשתני המחלקה.

**פונקציות:**

**RenderImage** - פונקציה שמקבלת את נקודות החיתוך ומוצאת את הצבע המתאים לכל פיקסל.

לכל נקודה על המסך מבצעים את האלגוריתם הבא:

1. שולחים קרניים מהמצלמה לכל נקודה על המסך, ואת הקרן שומרים במשתנה מטיפוס ray.
2. ב. על מנת למצוא את הקרן משתמשים בפונקציית **constructRayThroughPixel** השייכת למחלקת מצלמה.
3. ג. קוראים לפונקציה **findClosestIntersection** שמחזירה את נקודת החיתוך הקרובה ביותר, אם נמצאת.
4. אם אין נקודות חיתוך, הכנס למשתנה **ImageWriter** את הנקודה הנתונה ואת צבע הרקע.
5. אם יש נקודת חיתוך מצא את הצבע שצריך להיות בנקודה ע"פ **calcColor**, ומכניסים את הנקודה וצבעה למשתנה **ImageWriter**.

**findClosestIntersection -** מקבלת טיפוס ממשתנה ray, מייצגת מיקום במפה של נקודת החיתוך הקרובה ביותר.  
הפונקציה מייצרת Map ע"י הפונקציה getSceneRayIntersections, היא שומרת בkey את הצורות הגיאומטריות שנחתכות עם הקרן.  
שומרת ב value רשימה המונה נקודות חיתוך שלהן.  
כשיש נקודות חיתוך משתמשת בפונקציה getClosestPoint -שתחזיר את הנקודה הקרובה ביותר.

היא מחזירה: כשיש נקודות חיתוך, מחזירה ערך מטיפוס Entry<Geometry,Point3D>.  
כשאין נקודות חיתוך מחזירה Null.

**getSceneRayIntersections-** מקבלת טיפוס ממשתנה ray, מבצעת מעבר עם הקרן על כל הצורות הגיאומטריות שבסצנה.  
המעבר הוא בעזרת איטרטור של רשימת הצורות הגיאומטריות.  
בודקת אם יש נקודות חיתוך בין הקרן לצורות הגיאומטריות – שימוש בפונקציה findIntersections.

מחזירה רשימה מטיפוס Map<Geometry, List<Point3D>> רשימה של כל נקודות החיתוך של כל הצורות הגיאומטריות בסצנה עם הקרן הנתונה.

**printGrid –** היא יוצרת רשת, ע"י משתנה מסוג Int – מרווח של הרשת. עוברת על המסך בקפיצות לפי הפרמטר שקיבלה וצובעת את הרשת ע"פ הצבע המתאים לאורך ולרוחב. מחזירה משתנה מסוג imageWriter יש בו את הערכים של הרשת.

**writeToImage** – קוראת לפונקציה מתאימה במחלקה ImageWriter – מדפיסה את הנתונים שבimageWritet למסך.

**addColors** – מקבלת שני משתני color ומחברת את משתני Color והכנסתם למשתנה חדש, הפעולה קוראת ע"י לקיחת כל ערך Int של כל צבע וחיבורם, חיבור שיותר מ255 יש אתחול ל255 שזה המספר המקסימלי. מחזירה ערך מסוג Color.

**calcColor** - מקבלת שני משתנים אחד מטיפוס Geometry, והשני מטיפוס Point3d**.**

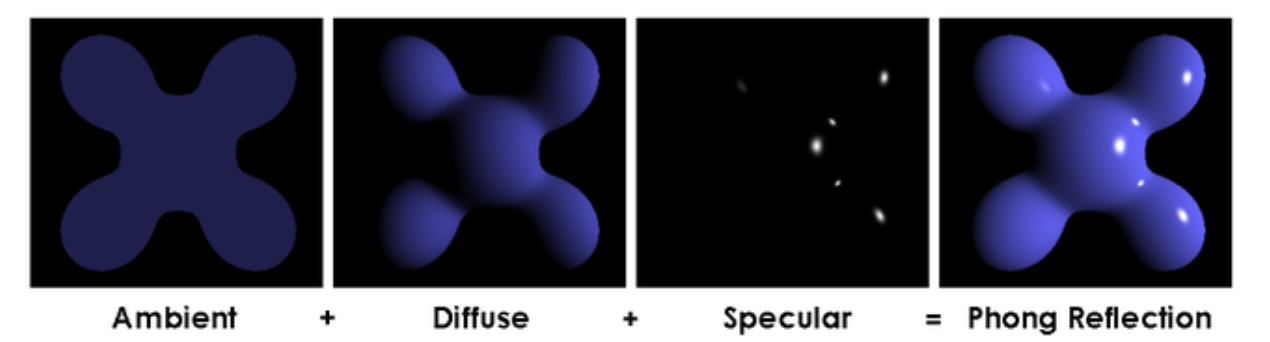
מספקת צבע סופי לכל נקודה, לפי כל הפרמטרים השונים**,** מחזירה ערך מסוג Color הצבע המתקבל.

**הנוסחא לקבלת צבע:**

נתינת צבע לכל נקודה נקבע לפי פרמטרים של שיטת פונג:

1. אור סביבתי – Ambiet Light
2. הצבע והחומר של העצם.
3. הפיזור של האור – diffuseLight – האור בהתחשב כל גופי התאורה.
4. הברק של האור – specularLight – האור בהתחשב כל גופי התאורה.

עבור כל התאורות בסצנה.



**האלגוריתם:**

אם רמת הרקורסיה של הפונקציה שווה לרמת הרקורסיה המקסימלית – מחזיר שחור.

אחרת – קולט למשתנה את הצבע של התאורה הכללית, של האור הסביבתי, ע"י הפונקציה getIntersity, נמצאת במחלקה AmbientLight.

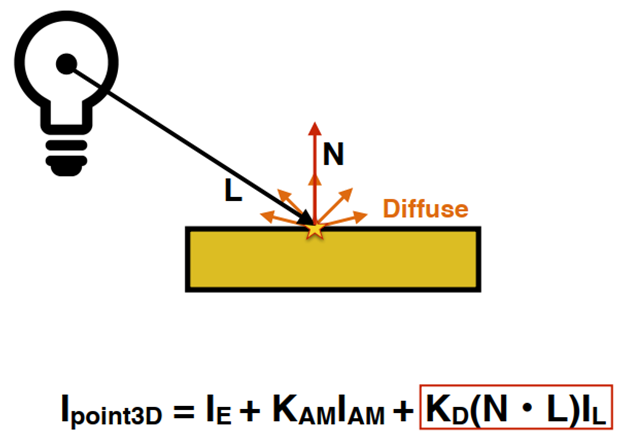
קולט את הצבע שהצורה עצמה מפיקה ע"י הפונקציה getEmmission שבמחלקת Geometry.עובר על כל התאורות בסצנה, אם האור לא מגיע לנקודה, תחזיר את הצבע שחושב עד עכשיו. אם האור מגיע, מחשב גם את הפיזור של האור בנקודה הרצויה של כל אור ע"י הפונקציה calcDffusiveComp, ואותו דבר לגבי הברק, רק שמשתמשים עם הפונקציה calcSpecularComp , לאחר מכן אם יש לקרן הנתונה קרן שמשתקפת ממנה קוראים לרקורסיה שוב ומעלים אותה ב1 וכך מחשבים את נקודות ההשתקפות בסצנה וכך עושים לקרניים שחודרות לצורה הגיאומטרית, מחברים את כל החישוב ומכניסים למשתנה Color.

**calcDiffusiveComp**

**תפקידה**-מחזירה את צבע הפיזור בנקודה. מקבלת ארבעה **משתנים**:

1. Kd – משתנה מטיפוס double , פרמטר קבוע שמכפיל את האור בהתאם לכל צורה.
2. Normal – משתנה מטיפוס Vector, מייצג את הנורמל של הצורה בנקודה.
3. L Vector שמייצג וקטור כיוון של התאורה.
4. LightIntesity – משתנה מטיפוס Color, מייצג את צבע התאורה.

**אלגוריתם**- הכפלת משתנה kd עם תוצאה של מכפלה סקלרית בין הנורמל לבין L -כיוון התאורה – זה הזווית של הכיוון לעומת הנורמל, וכך הפיזור יותר גדול.הכפלת תוצאה מסעיף קודם בצבע התאורה של גוף התאורה – ע"י לקיחת ערכי הint של RGB ומכפילים בערכי RGB השני, אם עובר את 255, אז מאותחל אוטומטית ל255.



**calcSpecularComp** פקידה להחזיר את הברק של הצורה הגיאומטרית בנקודה מסויימת לפי התאורה.

ישנם שישה **משתנים**:

1. Ks – משתנה מטיפוס double , פרמטר קבוע שמכפיל את האור בהתאם לכל צורה.
2. V – משתנה מטיפוס Vector, מייצג וקטור אנכי, לכיוון הקרן של התאורה לצורה הגיאומטרית.
3. Normal – משתנה מטיפוס Vector, מייצג את הנורמל של הצורה בנקודה.
4. L – משתנה מטיפוס Vector, שמייצג וקטור כיוון של התאורה.
5. Shininess – משתנה מטיפוס double , מייצג את הזוהר של הנקודה, ע"י זה שמעלה נתון מסוים בחזקה.

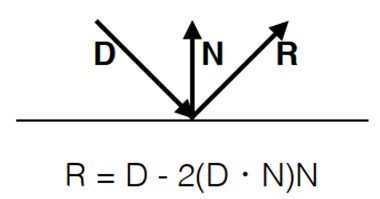
**LightIntesity** – משתנה מטיפוס Color , מייצג את צבע התאורה.

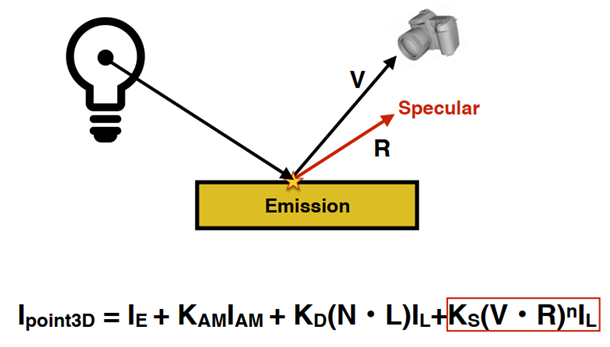
**אלגוריתם** –

הפונקציה מחשבת את המסגרת האדומה שיש באיור למעלה, המכפלה הסקלרית בין V לבין R, הזווית בין המצלמה לקרן החוזרת מהאור, זה מידול של העולם הפיזיקאלי של משהו מבריק = קרן החזרת לעין מהתאורה.

התוצאה עולה בחזקת ה – Shininess בגלל שבעולם הפיזקלי אור הברק יורד בצורה מאוד מהירה, וכל זה מוכפל באור, כשמכפילים את התוצאה, זה אומר שלוקחים את ערכי ה int של RGB ומכפילים בערכי הint של RGB השני, ערך שעובר את 255 מאותחל אוטומטית ל255.

חישוב פרמטר R;





**- Occluded**תפקידה לבדוק האם מקור אור מסוים מגיע לנקודה שעל צורה גיאומטרית.

מקבלת שלושה **משתנים**:

1. משתנה מטיפוס LightSource – מקור אור.
2. Point3D – נקודה שנחתכה ע"י הקרן מהמצלמה.
3. משתנה מטיפוס Geometry – צורה גיאומטרית שבה נמצאת הנקודה.

**אלגוריתם**: חישוב וקטור בין הנקודה לבין מקור האור, ובודק האם יש משהו חוסם בניהם.למניעת שגיאת floating point – מוציאים נורמל מכל נקודה שנמצאת על הצורה הגיאוטמרית. בצורות שטוחות אין נורמל, ולכן מוחקים מרשימת נקודות החיתוך את נקודות שלה עצמה, אם יש צורה גיאומטרית אחת שאינ שקופה שנמצאת ברשימת נקודות החיתוך, הפונקציה מחזירה true אחרת מחזירה false.

**constructReflectedRay-** תפקידה למצוא קרן המשתקפת מהצורה הגיאומטרית מהקרן הנתונה.

מקבלת שלושה **משתנים:**

* 1. Vector – נורמל של הצורה הנתונה בנקודה.
  2. Point3D – נקודה שעל הצורה.
  3. Ray – קרן אור לנקודה.

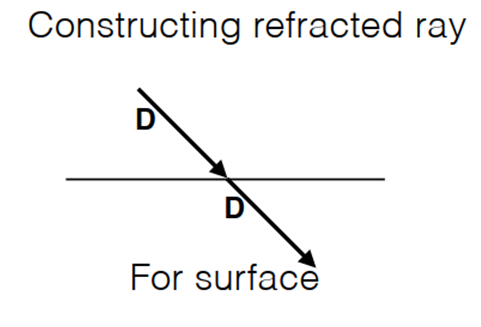
**אלגוריתם**: כמו לעיל בחישוב R.

**- constructRefractedRay**תפקידה למצוא קרן החודרת לתוך צורה גיאומטרית ע"י הקרן הנתונה, מחזירה את הקרן החודרת.

מקבלת שלושה **משתנים:**

1. Vector – נורמל של הצורה הנתונה בנקודה.
2. Point3D – נקודה שעל הצורה.
3. Ray – קרן אור לנקודה.

נוסחא:



TESTS

1. DiretionalLight –תאורת שמש
2. PointLight – תאורת נורה, נקודתית
3. SpotLight – תאורת פרוז'קטור.
4. ShadowTest – טסט צל, לפי פרמטרים של מיקום, זוית, צבע.

***תוצאות סופיות בשלב 3:***

***|***

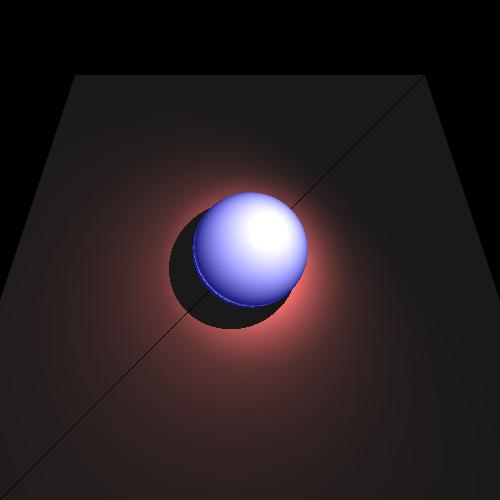
***|***

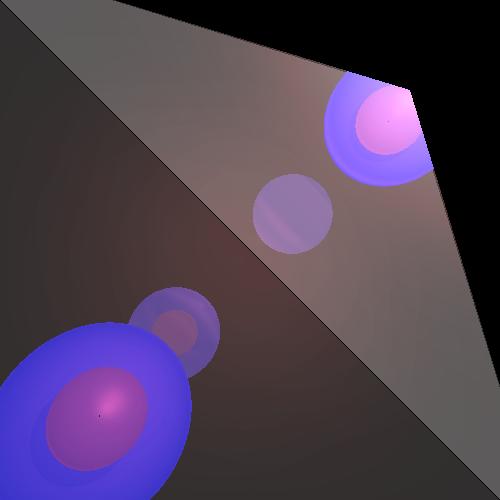
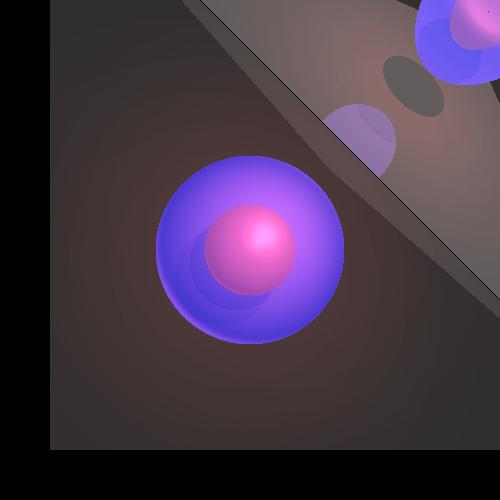
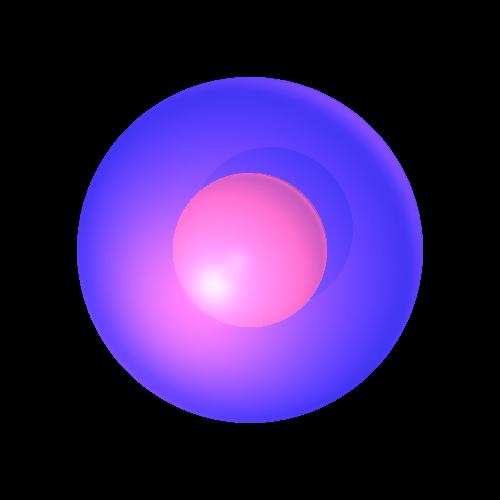
***|***

**V**

**Moodle’s Tests**

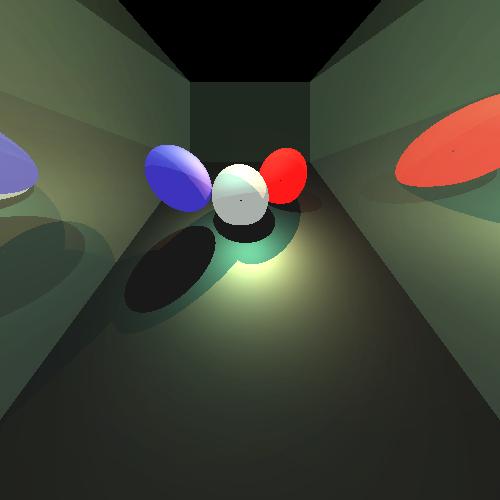
**Shadow Test:**

****

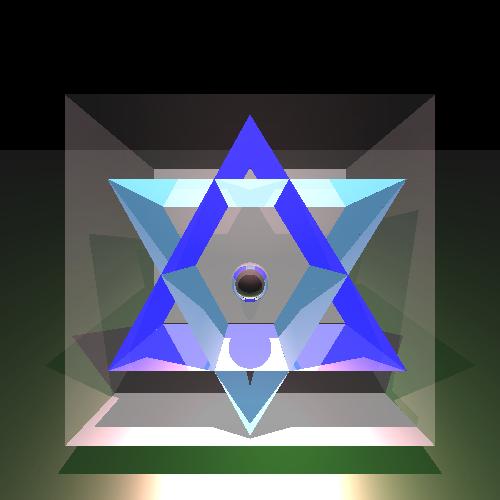
**Recursive Tests:**

**Our Tests**

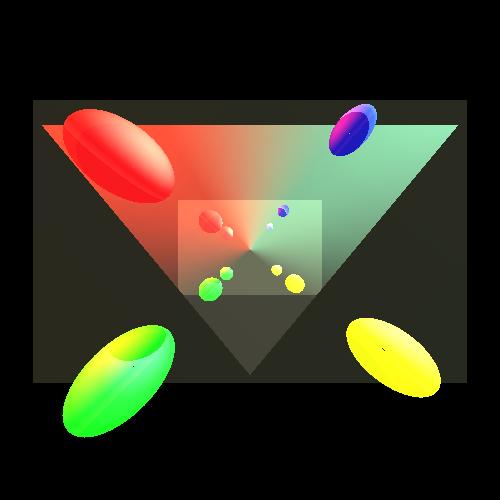
**PointLight:**

****

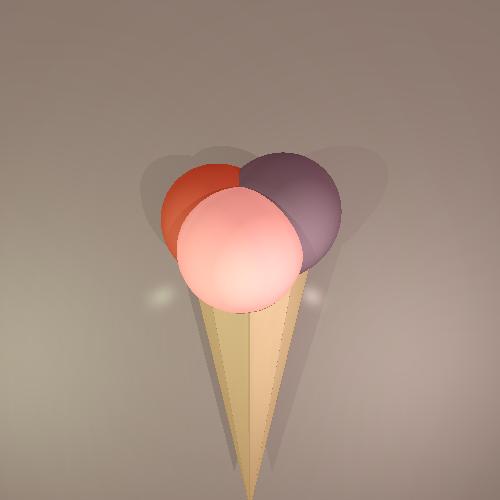
**PointLight + DirectionLight:**

****

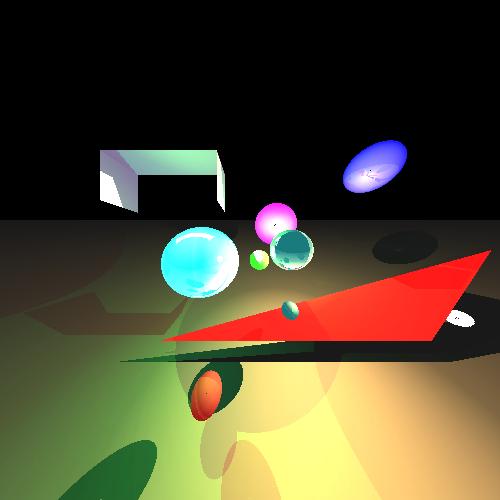
**SpotLight:**



**SpotLight:**

****

**DirectionLight:**

****

שיפור תמונה

**הבעיה שלנו:** זכוכית שקופה ולא מבריקה / משטחים מבריקים לחלוטין

**פתרון:** ריבוי קרני החזרה.Multiple reflection rays

וריבוי קרני בליעה .Multiple refraction rays

**נקודות למחשבה: פיזור אחיד בפיקסל ולא על קו או על ,Xהיכן ממוקם קוד יצירת  
הקרניים המרובות והיכן חישובי הצבע והמיצוע. לכלול את הקרן המרכזית, מיצוע נכון**