

מיני פרויקט במבוא להנדסת תוכנה

151055.3.5781

**Mini project in Introduction to Software
Engineering**

מגישות:

אנה זרביב- 340941244

קרן סממה- 336351366



תודה גדולה למרצה אליעזר גינסבורגר!

תוכן עניינים

מינפרויקט 1: שיפור תמונה

- 4.....הסבר הבעיה
- 5.....מימוש השיפור
- 5.....תוצאות עם טסט

מינפרויקט 2 : שיפור ביצועים

- 4.....הסבר הבעיה
- 5.....מימוש השיפור
- 5.....תוצאות עם טסט

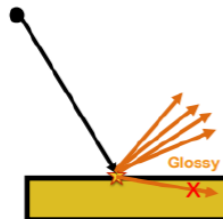
מיני פרויקט 1 : שיפור תמונה

Glossy Surface and Diffuse Glass

1. הסבר הבעיה :

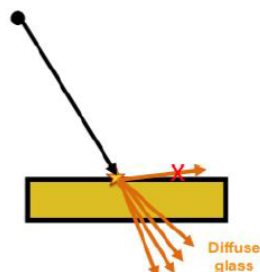
Glossy surface •

- הבעיה - שזכוכית משקפת לחלוטין
- הפתרון - במקום לשלוח קרן אחד של השתקפות נשלח יותר
- תוצאה - מה שיותר קרוב יראה חד ומה שרחוק יותר יראה מטושטש בגלל שהתפזרות הקרניים מתרחבת עם המרחק



Diffuse glass •

- הבעיה - משטחים שמבריקים באופן מושלם
- הפתרון - במקום לשלוח קרן אחד של שקיפות נשלח יותר
- תוצאה - מה שיותר קרוב יראה חד ומה שרחוק יותר יראה מטושטש בגלל שהתפזרות הקרניים מתרחבת עם המרחק



2. המימוש של השיפור

הוספנו פונקציה במחלקה Vector בשם **normalToVector** שמחשבת את הנורמל לווקטור. אצלנו מי שקוראת לפונקציה זאת תהיה הווקור של קרן השתקפות הראשון או קרן השקיפות הראשון שנוצר. בפונקציה, רוצים לאפס אחד הקואורדינטה ולהחליף סימן לאחרים אבל אם חלק הקואורדינטות אפסיות לא יכולות להיות כולם אפסיות לכן נאפס את הקואורדינטה הכי קטנה והסימן נחליף לאחד הגדולים.

```

} /**
 * This function helps us to calculate a normal vector to the vector that calls the function
 * @return a new vector
 */
public Vector normalToVector()
{
    double coordinate;

    if(this.getHead().getX()>0) //finding the smallest coordinate of the vector to replace it with 0
    {
        coordinate = this.getHead().getX();
    }
    else {
        coordinate = -this.getHead().getX();
    }

    if(Math.abs(this.getHead().getY())<coordinate)
    {
        coordinate=1;
        if(this.getHead().getY()>0)
            coordinate=this.getHead().getY();
        else
            coordinate=-this.getHead().getY();
    }
    if(Math.abs(this.getHead().getZ())<coordinate)
    {
        coordinate=2; //last coordinate that we are checking so no need to reassign coordinate
    }

    if(coordinate==0) { //x is the smallest
        return new Vector( x: 0, -this.getHead().getZ(), this.getHead().getY()).normalize();
    }
    if(coordinate==1) { //y is the smallest
        return new Vector(-this.getHead().getZ(), y: 0, this.getHead().getX()).normalize();
    }
    //z is the smallest
    return new Vector(this.getHead().getY(),-this.getHead().getX(), z: 0).normalize();
}

```

הוספנו במחלקה של קרן פונקציה בשם :
.createBeamOfRays

הפונקציה מייצרת אלומת קרניים לפי הקרן הראשית שהוא קרן שקיפות או קרן השתקפות. הפונקציה מקבלת כפרמטרים וקטור נורמל לראשית הקרן שקורא לפונקציה, מספר הקרניים שרוצים ליצור, והמרחק שאנחנו מחליטות ליצור מעגל וירטואלי שאנך לקרן שקורא לפונקציה. יצרנו שתי ווקטורים מאונכים לקרן הראשי : אחד בעזרת הפונקציה normalToVector והשני בעזרת ה Cross-product ביניהם. מפזרים נקודות רנדומליות בתוך המעגל ולפי זה מוצאים את הקרניים החדשות. בסוף לקחנו בחשבון קרניים מעל המשטח בשביל קרני השתקפות ומתחת למשטח בשביל קרני שקיפות.

```

1  /**
2   * This function creates a beam of rays
3   * @param normal of type Vector : normal vector to the head of the ray who calls the function |
4   * @param distance of type double : the distance between the point and the circle we are creating to find the beam
5   * @param numOfRays of type int : the number of rays that will be in the beam
6   * @return a list of all the rays in the beam
7   */
8   public List<Ray> createBeamOfRays(Vector normal, double distance, int numOfRays)
9   {
10      List<Ray> beam = new LinkedList<Ray>();
11      beam.add(this); // the original ray that calls the function - there has to be at least one ray
12      if (numOfRays == 1) // if no additional rays were requested here
13          return beam;
14
15      Vector w = this.getDir().normalToVector(); // finds a vector that is normal to the direction of the ray
16      // the cross product between the normal vector w and the direction gives a new normal vector to the direction of the ray
17      Vector v = this.getDir().crossProduct(w).normalize();
18
19      Point3D center = this.getPoint(distance); // the center of our circle is the distance requested from p0, the head of the ray that calls the function
20      Point3D randomP = Point3D.ZERO;
21      double xRandom, yRandom, random;
22
23      double randomRadiusValue = random(3, 6); // the radius will be in range: 3 < r < 6, and will have double values
24
25      for (int i = 1; i < numOfRays; i++) // starts from 1 because there has to be at least one ray (the original) and we already add it to the beam
26      {
27          xRandom = random(-1, 1); // random number [-1, 1]
28          yRandom = Math.sqrt(1 - Math.pow(xRandom, 2)); // toujours entre 0 et 1
29          random = random(-randomRadiusValue, randomRadiusValue);
30          if (xRandom == 0) // vector cannot be scaled with zero
31              randomP = center.add(w.scale(xRandom * randomRadiusValue)); //
32          if (yRandom == 0) // vector cannot be scaled with zero
33              randomP = center.add(v.scale(yRandom * randomRadiusValue)); // randomRadiusValue
34
35          Vector t = randomP.subtract(this.getP0()); // vector from the head of the original ray and the random point
36
37          double normalDotDir = AlignZero(normal.dotProduct(this.getDir()));
38          double normalDotT = AlignZero(normal.dotProduct(t));
39          if (normalDotDir * normalDotT > 0) // if they are both positive or both negative then we need to create a ray with the original p0 and t
40              beam.add(new Ray(this.getP0(), t));
41      }
42      return beam;
43  }

```

הוספנו במחלקת BasicRayTracer את המשתנים **_numOfRays** בשביל מספר הקרניים באלומת קרניים שרוצים ליצור ו**_rayDistance** כדי לדעת מאיזה מרחק ליצור מעגל וירטואלי. שני הפרמטרים עם ערך ברירת מחדל אפס כדי שלא נצטרך לתקן את כל הטסטים הקודמים.

```

/**
 * Parameters for ray tracing- glossy surface and diffuse glass - they are in class BasicRayTracer
 * because this class takes care of ray tracing.
 */
private int _numOfRays=0;
private double _rayDistance=0;
/**

/**
 * Set the distance between the intersection point and the circle
 * @param _rayDistance of type double
 */
public BasicRayTracer set_rayDistance(double _rayDistance) {
    if (_rayDistance < 0)
        throw new IllegalArgumentException("Distance cannot be negative");
    this._rayDistance = _rayDistance;
    return this;
}

/**
 * Get the distance we want between the intersection point and the circle
 * @return double for distance
 */

public double get_rayDistance() { return _rayDistance; }

/**
 * Set the number of rays that will be part of the beam
 * @param _numOfRays of type int : amount of rays that will be part of the beam
 */
public BasicRayTracer set_numOfRays(int _numOfRays) {
    if (_numOfRays < 0)
        throw new IllegalArgumentException("Number of rays cannot be negative");
    this._numOfRays = _numOfRays;
    return this;
}

/**
 * Get number of rays of the beam
 * @return number of rays that will be part of the beam
 */
public int get_numOfRays() { return _numOfRays; }

```

כדי לתמוך בשיפורים, בסוף שינינו את הפונקציה

calcGlobalEffects

כי היא מחשבת את האפקים של קרני שקיפות והשתקפות. הוספנו את היצירה של האלומת קרניים על שקיפות והשתקפות וביצענו קריאה רקורסיבית עבור כל קרן ששייך לאלומה והכנסנו את הצבע למשתנה זמני ואחרי שסיימנו את הקריאות הרקורסיביות חילקנו במספר הקרניים ואז הוספנו לצבע.

```

/**
 * This function takes account of global effects of interactions
 * between objects such as reflections and refractions of light.
 * This helps to be able to model transparent objects (with various
 * opacity levels) and reflecting surfaces such as mirrors.
 * @param geoPoint of type GeoPoint : the closest intersection point with the head of the ray from the camera
 * @param ray of type Ray : a ray from the camera
 * @param level the recursion level
 * @param k double - helps with recursion
 * @return color object from
 */
private Color calcGlobalEffects(GeoPoint geoPoint, Ray ray, int level, double k) {
    Color color = Color.BLACK;
    Color ReflectedColor = Color.BLACK;
    Color RefractedColor = Color.BLACK;
    Vector n = geoPoint._geometry.getNormal(geoPoint._point);
    Material material = geoPoint._geometry.getMaterial();
    List<Ray>beam1=new LinkedList<>(); // for beam of reflected ray
    List<Ray>beam2=new LinkedList<>(); // for beam of refracted ray

    double kkr = k * material.Kr;
    if (kkr > MIN_CALC_COLOR_K) { //if the reflection is bigger than the minimum of calc color

        Ray reflectionRay = constructReflectedRay(geoPoint,ray.getDir());
        //color = calcGlobalEffects(reflectionRay, level, material.Kr, kkr);
        if(this._numOfRays==0 || this._rayDistance<=0){
            beam1.add(reflectionRay);
        }
        else {
            beam1= reflectionRay.createBeamOfRays(geoPoint._geometry.getNormal(geoPoint._point), this.get_rayDistance(),this.get_numOfRays());
        }
        for(Ray r : beam1) // r = reflectedRay
        {
            ReflectedColor = ReflectedColor.add(calcGlobalEffects(r, level, material.Kr, kkr)); // //calls the recursion to find the rest of the color
        }
        if(beam1.size()>0) {
            color = color.add(ReflectedColor.reduce(beam1.size()));
        }
    }

    double kkt = k * material.Kt;
    if (kkt > MIN_CALC_COLOR_K) { //if the refraction is bigger than the minimum of calc color
        Ray refractionRay = constructRefractedRay(geoPoint,ray.getDir());
        //color = color.add(calcGlobalEffects(refractionRay, level, material.Kt, kkt));
        if(this._numOfRays==0 || this._rayDistance<=0)
            beam2.add(refractionRay);
        else{
            beam2= refractionRay.createBeamOfRays(geoPoint._geometry.getNormal(geoPoint._point), this.get_rayDistance(),this.get_numOfRays());
        }
        for(Ray r : beam2) // r = refractedRay
        {
            RefractedColor = RefractedColor.add(calcGlobalEffects(r, level, material.Kt,kkt)); // //calls the recursion to find the rest of the color
        }
        if(beam2.size()>0) {
            color = color.add(RefractedColor.reduce(beam2.size()));
        }
    }

    return color;
}

```

3. התוצאות :

הטסט שכתבנו כדי להראות את השיפורים. כל פעם שינינו מספר קרניים בעזרת
 .Set_numOfRays()
 ושינו את השם של התמונה.

```

134 public void finalProject1() {
135
136     Scene scene2 = new Scene( name: "Test scene")//
137     .setAmbientLight(new AmbientLight(new Color( r: 255, g: 255, b: 255), Ka: 0.15)) //
138     .setBackground(new Color( r: 0, g: 0, b: 0));
139
140
141     scene2.geometries.add(new Sphere( radius: 20, new Point3D( x: 125, y: -36, z: 50))// violet
142     .setEmission(new Color( r: 108, g: 2, b: 139))
143     .setMaterial(new Material().setKd(0.3).setKs(0.2).setShininess(5).setKt(0).setKr(0)));
144
145     scene2.geometries.add(new Sphere( radius: 20, new Point3D( x: 125, y: 100, z: 50))// orange
146     .setEmission(new Color( r: 231, g: 62, b: 1))
147     .setMaterial(new Material().setKd(0.25).setKs(0.3).setShininess(5).setKt(0.22).setKr(0)));
148
149     scene2.geometries.add(new Sphere( radius: 25, new Point3D( x: 0, y: -130, z: -90))// bleu milieu en bas
150     .setEmission(new Color( r: 0, g: 128, b: 128))
151     .setMaterial(new Material().setKd(0.25).setKs(0.3).setShininess(5).setKt(0).setKr(0.3)));
152
153
154     scene2.geometries.add(new Sphere( radius: 25, new Point3D( x: 0, y: -130, z: +90))//vert milieu dessus
155     .setEmission(new Color( r: 0, g: 128, b: 85)) //0 128 85
156     .setMaterial(new Material().setKd(0.25).setKs(0.3).setShininess(5).setKt(0).setKr(0.3)));
157
158     scene2.geometries.add(new Sphere( radius: 50, new Point3D( x: -63, y: +30, z: -200))//BIG BLUE
159     .setEmission(new Color( r: 15, g: 5, b: 107))
160     .setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0)));
161
162
163     scene2.geometries.add(new Sphere( radius: 50, new Point3D( x: 48, y: 30, z: -200))// BIG JAUNE
164     .setEmission(new Color( r: 212, g: 175, b: 55))
165     .setMaterial(new Material().setKd(0.3).setKs(0.2).setShininess(5).setKt(0).setKr(0.1)));
166
167     scene2.geometries.add(new Sphere( radius: 20, new Point3D( x: -180, y: 90, z: -70))// red
168     .setEmission(new Color( r: 255, g: 0, b: 0))
169     .setMaterial(new Material().setKd(0.3).setKs(0.2).setShininess(5).setKt(0).setKr(0)));
170
171     scene2.geometries.add(new Sphere( radius: 20, new Point3D( x: -180, y: -36, z: -70))// GREEN a gauche
172     .setEmission(new Color( r: 34, g: 80, b: 30))
173     .setMaterial(new Material().setKd(0.3).setKs(0.4).setShininess(5).setKt(0.22).setKr(0)));
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```


The image displays a Blender 2.80.2 interface with a Cycles render of a complex, multi-faceted geometric object. The object is composed of numerous flat, triangular and quadrilateral faces, creating a faceted, crystalline appearance. It is rendered in a dark, monochromatic style with a light gray background. The interface includes a top status bar with 'Final project 1 ray.png' and 'Final project 5 rays.png'. The left sidebar shows the 'Properties' panel for the 'Material' property, with 'Color' set to (0.0, 0.0, 0.0) and 'Emission' set to (0.0, 0.0, 0.0). The right sidebar shows the 'Material' property for the 'Material' property, with 'Color' set to (0.0, 0.0, 0.0) and 'Emission' set to (0.0, 0.0, 0.0). The bottom status bar shows 'Final project 1 ray.png' and 'Final project 5 rays.png'.

```

project 1 rays.png x RenderTests.java x Main.java x final project 10 rays.png x Ray.java x BasicRayTracer.java x final project 5 rays.png x finalProject.png x Vec
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(-125, -62.5, -250), new Point3D(-187.5, -62.5, -250),
new Point3D(-187.5, -125, -250), new Point3D(-125, -125, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.1).setKs(0.1).setShininess(5).setKt(0).setKr(0.7));

scene2 geometries.add(new Polygon(new Point3D(-187.5, -62.5, -250), new Point3D(-250, -62.5, -250),
new Point3D(-250, -125, -250), new Point3D(-187.5, -125, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

//-----line 7
scene2 geometries.add(new Polygon(new Point3D(250, -125, -250), new Point3D(187.5, -125, -250),
new Point3D(187.5, -62.5, -250), new Point3D(250, -62.5, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(187.5, -125, -250), new Point3D(125, -125, -250),
new Point3D(125, -62.5, -250), new Point3D(187.5, -62.5, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(125, -125, -250), new Point3D(62.5, -125, -250),
new Point3D(62.5, -62.5, -250), new Point3D(125, -62.5, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(62.5, -125, -250), new Point3D(0, -125, -250),
new Point3D(0, -62.5, -250), new Point3D(62.5, -62.5, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(0, -125, -250), new Point3D(-62.5, -125, -250),
new Point3D(-62.5, -62.5, -250), new Point3D(0, -62.5, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(-62.5, -125, -250), new Point3D(-125, -125, -250),
new Point3D(-125, -62.5, -250), new Point3D(-62.5, -62.5, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(-125, -125, -250), new Point3D(-187.5, -125, -250),
new Point3D(-187.5, -62.5, -250), new Point3D(-125, -62.5, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(-187.5, -125, -250), new Point3D(-250, -125, -250),
new Point3D(-250, -62.5, -250), new Point3D(-187.5, -62.5, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

//-----line 8
scene2 geometries.add(new Polygon(new Point3D(250, -187.5, -250), new Point3D(187.5, -187.5, -250),
new Point3D(187.5, -250, -250), new Point3D(250, -250, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(187.5, -187.5, -250), new Point3D(125, -187.5, -250),
new Point3D(125, -250, -250), new Point3D(187.5, -250, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(125, -187.5, -250), new Point3D(62.5, -187.5, -250),
new Point3D(62.5, -250, -250), new Point3D(125, -250, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(62.5, -187.5, -250), new Point3D(0, -187.5, -250),
new Point3D(0, -250, -250), new Point3D(62.5, -250, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(0, -187.5, -250), new Point3D(-62.5, -187.5, -250),
new Point3D(-62.5, -250, -250), new Point3D(0, -250, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(-62.5, -187.5, -250), new Point3D(-125, -187.5, -250),
new Point3D(-125, -250, -250), new Point3D(-62.5, -250, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(-125, -187.5, -250), new Point3D(-187.5, -187.5, -250),
new Point3D(-187.5, -250, -250), new Point3D(-125, -250, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

scene2 geometries.add(new Polygon(new Point3D(-187.5, -187.5, -250), new Point3D(-250, -187.5, -250),
new Point3D(-250, -250, -250), new Point3D(-187.5, -250, -250))
.setEmission(new Color(0, 0, 0)).setMaterial(new Material().setKd(0.2).setKs(0.2).setShininess(5).setKt(0).setKr(0.5));

//-----
scene2 lights.add(new DirectionalLight(new Color(240, 240, 240), new Vector(0, 0, -1, 0));
scene2 lights.add(new PointLight(new Color(170, 170, 170), new Point3D(-60, -60, -60));
scene2 lights.add(new PointLight(new Color(170, 170, 170), new Point3D(60, 60, -60));
scene2 lights.add(new Spotlight(new Color(250, 250, 250), new Point3D(0, 0, -30), new Vector(0, 0, 1, 0) //
.setKl(0.5).setKs(0.5));

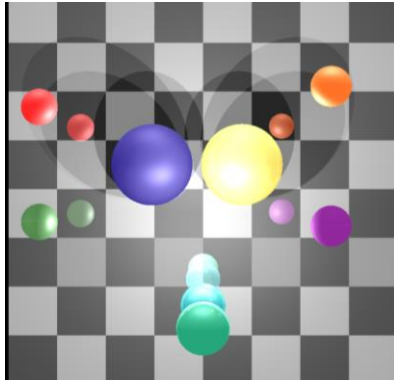
ImageWriter imageWriter = new ImageWriter("imageName: \"final project 50 rays\", 1000, 1000);

Render render = new Render() //
.setImageWriter(imageWriter) //
.setCamera(camera) //
.setMultithreading(3)
.setRayTracer(new BasicRayTracer(scene2).set_maxDistance(1).set_numOfRays(50));

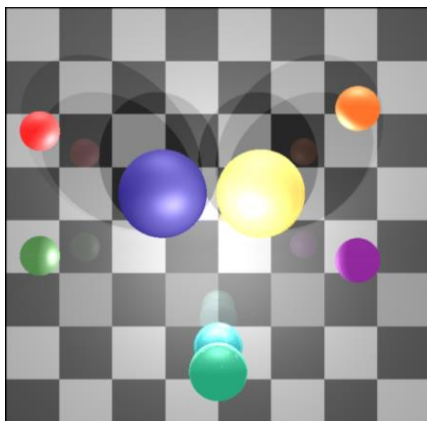
render.renderImage();
render.writeToImage();
}

```

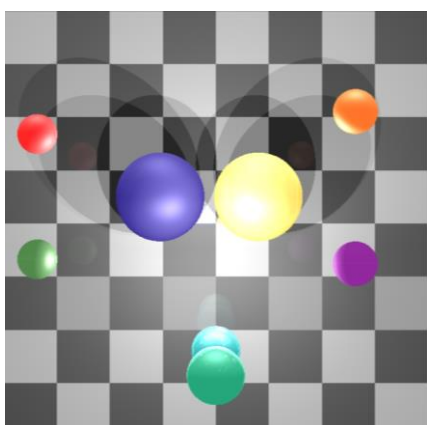
בלי שיפור קרן 1 : 14s 283 ms



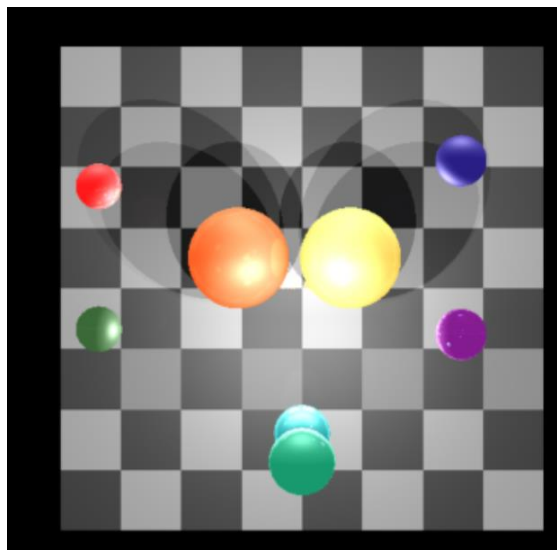
שיפור עם 5 קרניים : 23 s 201 ms



שיפור עם 10 קרניים : 59s 595 ms



שיפור עם 50 קרניים : 45 min 42s



מיני פרויקט 2 : שיפור ביצועים

Boundary Volume Hierarchy (BVH)

1. הספר הבעיה

מתחילת הפרוייקט , זרקנו המון קרניים וכל קרן בדקה חיתוך עם כל גאומטריות בסצנה , וזה בזבז ממש רציני של זמן ריצה. לכן כדי לשפר אותו ניתן להשתמש בשיטת עץ היררכי, BVH – boundary volume hierarchy . אנחנו עוטפים מספר גאומטריות בקופסא, ובודקים האם הקרן פוגעת בקופסא . אם היא פוגעת, ניכנס ונבדוק אם היא חותכת את הגאומטריות שבפנים . אבל אם היא לא פוגעת בה , היא לא תיכנס בה בכלל.

2. מימוש השיפור

שינוי של הממשק
Intersectable

למחלקה אבסטרקטית שהוספנו בה מחלקה פנימית כדי ליצור קופסא מינימלית.

הקופסא מיוצגת על ידי נקודה מינימלית ונקודה מקסימלית. הוספנו גם קריאה לבנאי ברירת מחדל בשביל שכל צורה יקבל קופסה.

הוספנו שדה `_setBoundingBox`

עם פעולה של סטר כדי להפעיל שיפור לטסטים וערך ברירת מחדל שווה לשקר כדי לא לדרוס את הטסטים שלנו.

```

13 public abstract class Intersectable{
14
15
16 /**
17  * _setBoundingBox is a parameter for BVH feature
18  * It is true if we want to use a bounding box for improvement
19  */
20 public boolean _setBoundingBox = false;
21
22 /**
23  * Setter function for _setBoundingBox
24  * @param setBoundingBox of type boolean
25  */
26 public void setBoundingBox(boolean setBoundingBox) { _setBoundingBox = setBoundingBox; }
27
28
29 /**
30  * A static internal helper class to create a Box. A box is represented by a minimal point and a maximal point.
31  */
32
33 public static class Box {
34     //makes them the opposite of what they should be so we can build boxes by checking if there is a bigger max or smaller min
35     public double _maxX = Double.NEGATIVE_INFINITY;
36     public double _minX = Double.POSITIVE_INFINITY;
37     public double _maxY = Double.NEGATIVE_INFINITY;
38     public double _minY = Double.POSITIVE_INFINITY;
39     public double _maxZ = Double.NEGATIVE_INFINITY;
40     public double _minZ = Double.POSITIVE_INFINITY;
41 }
42
43 // For each Intersectable shape, it calls to the default constructor to build a box
44 protected Box box = new Box();
45
46

```

פונקציה אבסטרקטית SetBoundingBox
שכל צורה יממש כדי ליצור קופסה מינימלית.

```

79 /**
80  * a function that every Intersectable geometry- should have in order to create
81  * a bounding box
82  */
83 abstract public void setBoundingBox();
84
85

```

הוספנו פונקציה בוליאנית שבודקת אם קרן חותכת קופסה.

```

/**
 * This function checks if the ray intersects the box around a shape.
 * @param ray of type Ray
 * @return true if there is an intersection, otherwise return false.
 */
public boolean isIntersectionWithBox(Ray ray) {

    Point3D start = ray.getP0();

    double start_X = start.getX();
    double start_Y = start.getY();
    double start_Z = start.getZ();

    Point3D direction = ray.getDir().getHead();

    double direction_X = direction.getX();
    double direction_Y = direction.getY();
    double direction_Z = direction.getZ();

    double max_t_for_X;
    double min_t_for_X;

    //If the direction_X is negative then the _min_X give the maximal value
    if (direction_X < 0) {
        max_t_for_X = (box._minX - start_X) / direction_X;
        // Check if the Intersectable is behind the camera
        if (max_t_for_X <= 0) return false;
        min_t_for_X = (box._maxX - start_X) / direction_X;
    }
    else if (direction_X > 0) {
        max_t_for_X = (box._maxX - start_X) / direction_X;
        if (max_t_for_X <= 0) return false;
        min_t_for_X = (box._minX - start_X) / direction_X;
    }
    else {
        if (start_X >= box._maxX || start_X <= box._minX)
            return false;
        else {
            max_t_for_X = Double.POSITIVE_INFINITY;
            min_t_for_X = Double.NEGATIVE_INFINITY;
        }
    }

    double max_t_for_Y;
    double min_t_for_Y;

    if (direction_Y < 0) {
        max_t_for_Y = (box._minY - start_Y) / direction_Y;
        if (max_t_for_Y <= 0) return false;
        min_t_for_Y = (box._maxY - start_Y) / direction_Y;
    }
    else if (direction_Y > 0) {

```

```

    max_t_for_Y = (box._maxY - start_Y) / direction_Y;
    if (max_t_for_Y <= 0) return false;
    min_t_for_Y = (box._minY - start_Y) / direction_Y;
}
else {
    if (start_Y >= box._maxY || start_Y <= box._minY)
        return false;
    else{
        max_t_for_Y = Double.POSITIVE_INFINITY;
        min_t_for_Y = Double.NEGATIVE_INFINITY;
    }
}

//Check the maximal and the minimal value for t
double temp_max = Math.min(max_t_for_Y,max_t_for_X);
double temp_min = Math.max(min_t_for_Y,min_t_for_X);
temp_min = Math.max(temp_min,0);

if (temp_max < temp_min) return false;

double max_t_for_Z;
double min_t_for_Z;

if (direction_Z < 0) {
    max_t_for_Z = (box._minZ - start_Z) / direction_Z;
    if (max_t_for_Z <= 0) return false;
    min_t_for_Z = (box._maxZ - start_Z) / direction_Z;
}
else if (direction_Z > 0) {
    max_t_for_Z = (box._maxZ - start_Z) / direction_Z;
    if (max_t_for_Z <= 0) return false;
    min_t_for_Z = (box._minZ - start_Z) / direction_Z;
}
else {
    if (start_Z >= box._maxZ || start_Z <= box._minZ)
        return false;
    else{
        max_t_for_Z = Double.POSITIVE_INFINITY;
        min_t_for_Z = Double.NEGATIVE_INFINITY;
    }
}

temp_max = Math.min(max_t_for_Z,temp_max);
temp_min = Math.max(min_t_for_Z,temp_min);

if (temp_max < temp_min) return false;

return true;
}

```

עזרה מאתרים:

[Introduction to Acceleration Structures \(Bounding Volume\) \(scratchapixel.com\)](http://scratchapixel.com/Introduction%20to%20Acceleration%20Structures%20(Bounding%20Volume))
[A Minimal Ray-Tracer: Rendering Simple Shapes \(Sphere, Cube, Disk, Plane, etc.\) \(Ray-Box Intersection\) \(scratchapixel.com\)](http://scratchapixel.com/A%20Minimal%20Ray-Tracer%3A%20Rendering%20Simple%20Shapes%20(Sphere,%20Cube,%20Disk,%20Plane,%20etc.)%20(Ray-Box%20Intersection))

במחלקה **Geometries**

שינינו את הבנאים ושיננו את פונקציית ההוספה של גיאומטריות כך שזה יחשב את הקופסאות על יותר מצורה אחת, על ידי קריאת לפונקציה **SetBoundingBox**

```

1  /**
2   * a function that every Intersectable geometry- should have in order to create
3   * a bounding box
4   * Set the bounding boxes between shapes - if there is a bigger max value of smaller min value for one of the coordinates than what we already have we will switch it
5   */
6   public void setBoundingBox()
7   {
8       double xMinLimit = this.box._minX; // going in the negative direction of x
9       double xMaxLimit = this.box._maxX; // going in the positive direction of x
10
11       double yMinLimit = this.box._minY; // going in the negative direction of y
12       double yMaxLimit = this.box._maxY; // going in the positive direction of y
13
14       double zMinLimit = this.box._minZ; // going in the negative direction of z
15       double zMaxLimit = this.box._maxZ; // going in the positive direction of z
16
17       for(Object geometry : _intersectables)
18       {
19           if(geometry instanceof Geometry)
20           {
21               Geometry geo = ((Geometry)geometry);
22
23               // setting x limits-----
24               if(geo.box._minX < xMinLimit)
25                   xMinLimit = geo.box._minX;
26
27               if(xMaxLimit < geo.box._maxZ)
28                   xMaxLimit = geo.box._maxZ;
29
30               // setting y limits-----
31               if(geo.box._minY < yMinLimit)
32                   yMinLimit = geo.box._minY;
33
34               if(yMaxLimit < geo.box._maxY)
35                   yMaxLimit = geo.box._maxY;
36
37               // setting z limits -----
38               if(geo.box._minZ < zMinLimit)
39                   zMinLimit = geo.box._minZ;
40
41               if(zMaxLimit < geo.box._maxZ)
42                   zMaxLimit = geo.box._maxZ;
43               //-----
44           }
45       }
46   }

```

```

1  //-----
2  //-----
3  //-----
4  //-----
5  //-----
6  //-----
7  //-----
8  //-----
9  //-----
10 //-----
11 //-----
12 //-----
13 //-----
14 //-----
15 //-----
16 //-----
17 //-----
18 //-----
19 //-----
20 //-----
21 //-----
22 //-----
23 //-----
24 //-----
25 //-----
26 //-----
27 //-----
28 //-----
29 //-----
30 //-----
31 //-----
32 //-----
33 //-----
34 //-----
35 //-----
36 //-----
37 //-----
38 //-----
39 //-----
40 //-----
41 //-----
42 //-----
43 //-----
44 //-----
45 //-----
46 //-----
47 //-----
48 //-----
49 //-----
50 //-----
51 //-----
52 //-----
53 //-----
54 //-----
55 //-----
56 //-----
57 //-----
58 //-----
59 //-----
60 //-----
61 //-----
62 //-----
63 //-----
64 //-----
65 //-----
66 //-----
67 //-----
68 //-----
69 //-----
70 //-----
71 //-----
72 //-----
73 //-----
74 //-----
75 //-----
76 //-----
77 //-----
78 //-----
79 //-----
80 //-----
81 //-----
82 //-----
83 //-----
84 //-----
85 //-----
86 //-----
87 //-----
88 //-----
89 //-----
90 //-----
91 //-----
92 //-----
93 //-----
94 //-----
95 //-----
96 //-----
97 //-----
98 //-----
99 //-----
100 //-----

```

התאמנו את הקוד בגאומטריות המתאימות שתהיה להם את האופציה
 לעבוד עם השיפור
Sphere במחלקת


```

/**
 * this function sets the values of the bounding box of the sphere
 */
@Override
public void setBoundingBox() {
    this.box._minX = _center.getX() - _radius; // x min
    this.box._maxX = _center.getX() + _radius; // x max
    this.box._minY = _center.getY() - _radius; // y min
    this.box._maxY = _center.getY() + _radius; // y max
    this.box._minZ = _center.getZ() - _radius; // z min
    this.box._maxZ = _center.getZ() + _radius; // z max
}

```

```

/**
 * This function helps to find the intersections between a ray and a sphere.
 *
 * @param ray of type Ray
 * @return A list of intersection points of type GeoPoint or null if there are no intersections
 */
@Override
public List<GeoPoint> findGeoIntersections(Ray ray, double maxDistance) {
    if(!_setBoundingBox || !_isIntersectionWithBox(ray))// if the ray does not intersect the bounding box
    {
        return null; // we will not calculate the intersection points with the sphere
    }

    Point3D p0 = ray.getP0(); // beginning point of the ray
    Vector v = ray.getDir(); // vector direction of the ray
    Point3D c = _center; // center of the sphere

    Vector u;
    try {

```

במחלקת Polygon

```

/**
 * this function sets the values of the bounding box of the polygon
 */
@Override
public void setBoundingBox() {
    //starting points
    box._minX = this._vertices.get(0).getX();
    box._maxX = this._vertices.get(0).getX();
    box._minY = this._vertices.get(0).getY();
    box._maxY = this._vertices.get(0).getY();
    box._minZ = this._vertices.get(0).getZ();
    box._maxZ = this._vertices.get(0).getZ();

    for (int j = 1; j < this._vertices.size(); j++) {
        if (this._vertices.get(j).getX() < box._minX)//checks for min x
            box._minX = this._vertices.get(j).getX();
        if (this._vertices.get(j).getY() < box._minY)//checks for min y
            box._minY = this._vertices.get(j).getY();
        if (this._vertices.get(j).getZ() < box._minZ)//checks for min z
            box._minZ = this._vertices.get(j).getZ();
        if (this._vertices.get(j).getX() > box._maxX)//checks for max x
            box._maxX = this._vertices.get(j).getX();
        if (this._vertices.get(j).getY() > box._maxY)//checks for max y
            box._maxY = this._vertices.get(j).getY();
        if (this._vertices.get(j).getZ() > box._maxZ)//checks for max z
            box._maxZ = this._vertices.get(j).getZ();
    }
}

```

```

d Run Tools Git Window Help Render.java - Polygon.java
etBoundingBox
ere.java x Tube.java x Plane.java x Polygon.java x
RenderTestsPictureBonus

/**
 * This function helps to find the intersection between a ray and a polygon.
 * First, we check intersection between the ray and the plane. If there is an intersection point,
 * we must update the geometry field from the GeoPoint originally received from the plane
 * to the value "this" related to the polygon
 *
 * @param ray of type Ray
 * @return a list of intersection points of type GeoPoint or null if there are no intersections
 */
@Override
public List<GeoPoint> findGeoIntersections(Ray ray, double maxDistance) {

    if(!_setBoundingBox || !_isIntersectionWithBox(ray))// if the ray does not intersect the bounding box
    {
        return null; // we will not calculate the intersection points with the polygon
    }

    List<GeoPoint> intersections = _plane.findGeoIntersections(ray, maxDistance);
    if (intersections == null) {
        return null;
    }
}

```

3. תוצאות

התמונה שלנו עם שליחת :

-קרן 1 : 10 s 302 ms

-5 קרניים : 16s 997 ms

-10 קרניים : 41s 371ms

סיכום

במהלך הפרוייקט, יצרנו גאומטריות, אלמנטים, פרימיטיביים, מצלמה, רנדר ועוד מחלקות רבות על מנת ליצור סצנות. השתדלנו לתת שמות משמעותיים למשתנים, הוספנו תיעוד להבנה. שיפרנו את התמונה בעזרת

Glossy surface and Diffuse glass.

זמן הריצה השתפר משמעותית עם הוספת שיפור של BVH.