



UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ESCUELA PROFESIONAL DE
CIENCIA DE LA COMPUTACIÓN

PRIMER EXÁMEN DE ESTRUCTURA
DE DATOS AVANZADO - OCTREE

PRIMER EXÁMEN DE ESTRUCTURA DE
DATOS AVANZADO - OCTREE

Alumno :

Nicoll del Rosario Aza

Mamani

Miguel Alexander Herrera

Cooper

Profesor:

Vicente Enrique Machaca

Arceda

2 de octubre de 2019

Índice

1. Introducción	2
1.1. Usos Comunes	2
1.2. La subdivisión del Espacio	3
1.3. Algoritmo de los Árboles Octales	3
2. Implementación de Octree	4
3. Conclusiones	9
4. Referencias	10

1. Introducción

Un octree es una estructura de datos en árbol en la que cada nodo interno tiene exactamente ocho hijos . Octrees son los más utilizados para dividir un espacio tridimensional por recursivamente subdividir en ocho octantes. Los octrees son el análogo tridimensional de los cuadrúpedos . El nombre está formado por oct + tree , pero tenga en cuenta que normalmente se escribe .octree con solo una "t". Los octrees a menudo se usan en gráficos 3D y motores de juegos 3D .

Cada nodo en un octree subdivide el espacio que representa en ocho octantes . En un octree de región de punto (PR), el nodo almacena un punto tridimensional explícito , que es el "centro" de la subdivisión para ese nodo; el punto define una de las esquinas para cada uno de los ocho hijos. En un octree basado en matriz (MX), el punto de subdivisión es implícitamente el centro del espacio que representa el nodo. El nodo raíz de un octree PR puede representar un espacio infinito; El nodo raíz de un árbol MX debe representar un espacio limitado finito para que los centros implícitos estén bien definidos. Tenga en cuenta que los octrees no son lo mismo que los árboles k-d : los árboles k-d se dividen a lo largo de una dimensión y los octrees se dividen alrededor de un punto. También k? ¿Cómo? '. K-d los árboles son siempre binarios, que no es el caso de los octrees. Al utilizar una búsqueda de profundidad, los nodos deben atravesarse y solo deben verse las superficies requeridas.

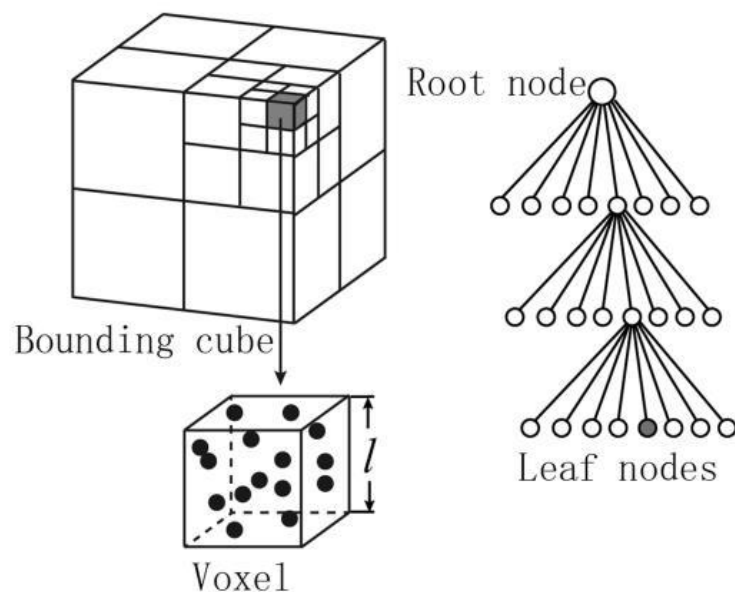
1.1. Usos Comunes

- Nivel de representación de detalle en gráficos 3D por computadora.
- Indexación espacial.
- Búsqueda de vecino más cercano.
- Detección eficiente de colisiones en tres dimensiones.
- Ver el sacrificio de Frustum.
- Método rápido multipolo.
- Rejilla no estructurada.
- Análisis de elementos finitos.
- Escaso voxel octree.
- Estimación del estado.
- Establecer estimación.

1.2. La subdivisión del Espacio

Al tratar de describir, un objeto mediante todo los voxels que le pertenecen, cuando cada cubo puede ser descrito por sus esquinas, se generan grandes listas de datos. La enumeración completa de los modelos hechos por descomposición tiene muchas ventajas, tales como su simplicidad, su generalidad y permiten un gran número de algoritmos. Sin embargo, el consumo de memoria es alto; se puede cambiar la división regular del espacio por algo más eficiente, una división adaptativa.

A este esquema de trabajo basado en la subdivisión del espacio se logra por simple observación de la rejilla descriptiva de la enumeración completa de un espacio 3D, donde muchas veces un cubo blanco tiene de vecinos otros cubos del mismo tipo "1". Al observar la combinación y codificarla en modelos de datos comparativamente similares. podemos reducir la cantidad de memoria utilizada por dicha rejilla.



1.3. Algoritmo de los Árboles Octales

Consiste en la metodología sistemática para generar y desplegar el modelo, donde la enumeración de atrás hacia adelante permite que los nodos sean listados, desplegando del más lejano al más cercano, desde un punto de VISIÓN, luego sus tres vecinos, en cualquier orden y así recursivamente.

Un modelador de sólidos con las funciones completas debe incluir los algoritmos que realicen las tareas siguientes:

- Generador del árbol, crea el octree desde las primitivas parametrizar o desde

otros tipos de modelos geométricos.

- Operaciones de Conjunto, que pueden llevarse a cabo entre dos octrees, con un espacio idéntico de interés, con el cálculo del octree resultante dado por la unión, intersección o diferencia de conjunto de los argumentos.
- Operaciones geométricas, que se hacen sobre un octree y calcular un nuevo octree como resultado de una traslación, rotación o escalamiento de los objetos modelados. Otro tipo de operación geométrica calcula el nuevo octree obtenido a partir de operaciones como las transformaciones de perspectivas. Algunos de estos problemas requieren un algoritmo complejo, muchas veces no intuitivo. La traslación de un octree es una operación bastante compleja y difícil.
- Procedimiento de análisis, calcula ciertas propiedades del objeto tales como su volumen o área de superficie del octree; un procedimiento de componentes conectado, este análisis es una operación muy compleja.
- Generador de desplegado, crea la imagen gráfica del objeto modelado por el octree.

2. Implementación de Octree

- Implementación del OCTREE en python y vtk

```
1     import vtk
2
3     puntos=[]#Arrays de Puntos
4     actors=[]#Puntos mapeados
5
6     def rect(x,y,z,w,h,p,color):
7         cube=vtk.vtkCubeSource()
8         cube.SetXLength(w)
9         cube.SetYLength(h)
10        cube.SetZLength(p)
11        cube.SetCenter(x,y,z)
12
13        cubeMapper=vtk.vtkPolyDataMapper()
14        cubeMapper.SetInputConnection(cube.GetOutputPort())
15
16        cubeActor=vtk.vtkActor()
17        cubeActor.GetProperty().SetColor(color[0],color[2],color[1])
18        cubeActor.GetProperty().SetOpacity(0.5)
```

```
19     cubeActor.SetMapper(cubeMapper)
20
21     actors.append(cubeActor)
22
23 def punto(x,y,z,color):
24     cube=vtk.vtkSphereSource()
25     cube.SetRadius(6.0)
26     cube.SetCenter(x,y,z)
27
28     cubeMapper=vtk.vtkPolyDataMapper()
29     cubeMapper.SetInputConnection(cube.GetOutputPort())
30
31     cubeActor=vtk.vtkActor()
32     cubeActor.GetProperty().SetColor(color[0],color[0],color[2])
33     cubeActor.SetMapper(cubeMapper)
34
35     puntos.append(cubeActor)
36
37 def Pantalla():
38     ren = vtk.vtkRenderer()
39
40     for act in puntos:
41         ren.AddActor(act)
42
43     for act in actors:
44         ren.AddActor(act)
45
46     renWin = vtk.vtkRenderWindow()
47     renWin.AddRenderer(ren)
48     renWin.SetSize(600, 600)
49     iren = vtk.vtkRenderWindowInteractor()
50     iren.SetRenderWindow(renWin)
51     ren.SetBackground(0,0,0)
52     renWin.Render()
53     iren.Start()
54
55 class Point:
56     def __init__(self, x, y, z):
57         self.x = x
58         self.y = y
59         self.z = z
60
61 class Rectangulo:
62     def __init__(self, x, y, z, w, h, p):
63         self.x = x
```

```
64         self.y = y
65         self.z = z
66         self.w = w
67         self.h = h
68         self.p = p
69     def cotiene(self, point):
70         return point.x >= self.x - self.w and point.x <= self.x +
self.w and point.y >= self.y - self.h and point.y <= self.y +
self.h and point.z >= self.z - self.p and point.z <= self.z +
self.p
71     def intersect(self, rango):
72         return not (rango.x-rango.w>self.x+self.w or rango.x+rango
.w<self.x-self.w or rango.y-rango.h>self.y+self.h or rango.y+
rango.h<self.y-self.h or rango.z-rango.p >self.z+self.p or
rango.z+rango.p<self.z-self.p)
73
74
75 class Octree:
76     def __init__(self, perimetro, n, color=[0,0,1]):
77         self.perimetro = perimetro
78         self.capacidad = n
79         self.points = []
80         self.divided = False
81         self.color=color
82
83     def Subdividir(self):
84         x = self.perimetro.x
85         y = self.perimetro.y
86         z = self.perimetro.z
87         w = self.perimetro.w/2
88         h = self.perimetro.h/2
89         p = self.perimetro.p/2
90
91         noup = Rectangulo(x-w,y+h,z+p,w,h,p);
92         neup = Rectangulo(x+w,y+h,z+p,w,h,p);
93         soup = Rectangulo(x-w,y-h,z+p,w,h,p);
94         seup = Rectangulo(x+w,y-h,z+p,w,h,p);
95
96         nodw = Rectangulo(x-w,y+h,z-p,w,h,p);
97         nedw = Rectangulo(x+w,y+h,z-p,w,h,p);
98         sodw = Rectangulo(x-w,y-h,z-p,w,h,p);
99         sedw = Rectangulo(x+w,y-h,z-p,w,h,p);
100
101         self.sonNOup = Octree(noup, self.capacidad, [noup.x/254,noup
.y/250, self.perimetro.p-noup.z/236])
```

```
102         self.sonNEup = Octree(neup, self.capacidad, [neup.x/254, neup
103         .y/250, self.perimetro.p-neup.z/236])
104         self.sonSOup = Octree(soup, self.capacidad, [soup.x/254, soup
105         .y/250, self.perimetro.p-soup.z/236])
106         self.sonSEup = Octree(seup, self.capacidad, [seup.x/254, seup
107         .y/250, self.perimetro.p-seup.z/236])
108
109         self.sonNodw = Octree(nodw, self.capacidad, [nodw.x/254, nodw
110         .y/250, self.perimetro.p-nodw.z/236])
111         self.sonNEdw = Octree(nedw, self.capacidad, [nedw.x/254, nedw
112         .y/250, self.perimetro.p-nedw.z/236])
113         self.sonSOdw = Octree(sodw, self.capacidad, [sodw.x/254, sodw
114         .y/250, self.perimetro.p-sodw.z/236])
115         self.sonSEdw = Octree(sedw, self.capacidad, [sedw.x/254, sedw
116         .y/250, self.perimetro.p-sedw.z/236])
117
118         self.divided = True
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
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
```



```
140         if(self.divided == True):
141             self.sonNOup.query(rango , found)
142             self.sonNEup.query(rango , found)
143             self.sonSOup.query(rango , found)
144             self.sonSEup.query(rango , found)
145             self.sonNodw.query(rango , found)
146             self.sonNEdw.query(rango , found)
147             self.sonSOdw.query(rango , found)
148             self.sonSEdw.query(rango , found)
149
150     def Mostrar(self):
151         rect(self.perimetro.x, self.perimetro.y, self.perimetro.z ,
self.perimetro.w*2, self.perimetro.h*2, self.perimetro.p*2, self.
color)
152         if(self.divided):
153             self.sonNOup.Mostrar()
154             self.sonNEup.Mostrar()
155             self.sonSOup.Mostrar()
156             self.sonSEup.Mostrar()
157             self.sonNodw.Mostrar()
158             self.sonNEdw.Mostrar()
159             self.sonSOdw.Mostrar()
160             self.sonSEdw.Mostrar()
161         for p in self.points:
162             punto(p.x, p.y, p.z, self.color)
163
164 from random import*
165
166
167 def main():
168     perimetro = Rectangulo (200,200,200,200,200,200)
169     ot= Octree(perimetro,8)
170     for i in range(0,77):
171         val1 = randrange(400)
172         val2 = randrange(400)
173         val3 = randrange(400)
174         p = Point(val1 , val2 , val3)
175         ot.Insertar(p)
176     ot.Mostrar()
177     rango = Rectangulo(randrange(200) ,randrange(200) ,randrange
(200) ,randrange(100) ,randrange(100) ,randrange(100))
178     rect(rango.x, rango.y, rango.z , rango.w*3, rango.h*3, rango.p
*3 , [255,0,0])
179
180     rango_punto= []
```

```

181 ot.query(rango , rango_punto)
182 for it in rango_punto:
183     punto(it.x, it.y, it.z, [255,0,0])
184 print(rango_punto)
185 Pantalla()
186 main()
187

```

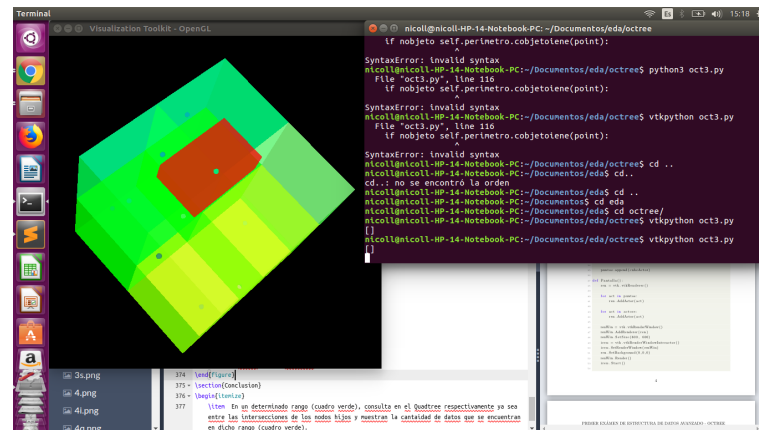


Figura 1: Prueba con 10 elementos

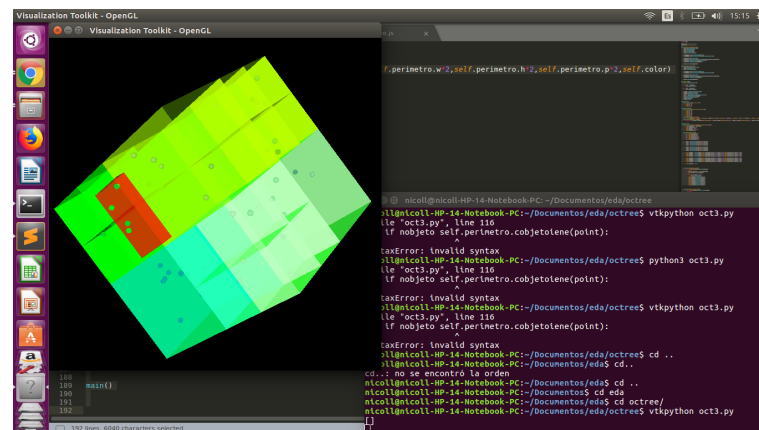


Figura 2: Prueba con 77 elementos

3. Conclusiones

- En un determinado rango (cuadro verde), consulta en el Quadtree respectivamente ya sea entre las intersecciones de los nodos hijos y muestran la cantidad de datos que se encuentran en dicho rango (cuadro verde).
- Es una forma de búsqueda que se da al Quadtree que es un respectivo árbol

- Gráficamente podemos visualizar a un octree como un cubo perfecto que representa al nodo raíz el cual encierra toda la geometría de la escena. Cada nodo está subdividido en ocho cubos más pequeños que representan a sus nodos hijo los cuales son llamados octantes. Cada uno de estos cubos también es subdividido en otros ocho octantes cada uno y así sucesivamente hasta llegar a una condición de término a la que típicamente se llega en el momento en que los octantes son de cierto tamaño o cuando no llegan a contener un cierto número mínimo de polígonos.
- Muchas de las operaciones que se pueden hacer, son ejecutadas de acuerdo al paradigma nodo-a-nodo, por ello, las propiedades integrales como volumen, centro de masa y momentos de inercia se pueden calcular por un simple algoritmo de árbol transversal.
- La idea de utilizar un octree para encontrar los nodos intransitables es para aprovechar la estructura y definición del octree para hacer un menor número de comparaciones al evaluar si un cubo o nodo en la malla esta ocupado o colisiona con un objeto de la escena, los nodos del octree que envuelvan a los objetos de la escena seran los nodos no transitables.
- Conversion and Integration of Boundary Representations with Octrees*

4. Referencias

- <https://www.gamedev.net/articles/programming/general-and-gameplay-programming/introduction-to-octrees-r3529/>
- Martti Mäntyla, An introduction to solid modeling, p. 68.
- Universidad de Oviedo - Esquemas de Modelado.
- http://catarina.udlap.mx/u_dla/tales/documentos/msp/mora_lma/capitulo2.pdf