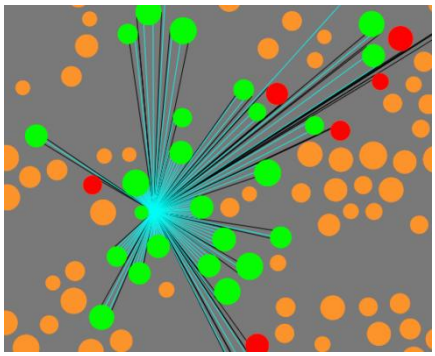
Visibility: Accurate calculation of the visibility of multiple Disks from arbitrary viewpoint.

Abhishek Venkatesh
venky@gatech.edu



CS:6491 3D Complexity, Project 2
Instructor Prof Jarek Rossignac

Abstract: In this paper we present a technique to compute the visibility of multiple disks from arbitrary view point. The location of disks is assumed to be fixed. One naive solution is shooting a ray from the viewpoint towards every direction (2*PI). But it suffers from inaccuracies which we point out in the paper. The method we describe in this paper computes accurate visibility and achieves this computation by shooting finite number of rays depending upon the number of disks. We also compare our approach with the naïve solution in terms of execution time and number of rays required to detect the exact visibility. Section I describes our approach, Section II compares it with the naïve approach, Section III gives results for multiple configurations of the Disks, and Section IV gives the conclusion and Future Work.

Keywords: Visibility**,** Disks, ray casting, tangents, occlusion

Introduction

Our goal is to calculate the accurate visibility of multiple disks from an arbitrary viewpoint. Visibility: If any portion of the disk is visible from the viewpoint we mark that disk as visible. Shooting sampled rays from the view point towards every direction has been proposed by Rossignac [1]. However, it is not accurate if the ray samples are too low or not enough. Another approach is to use the hardware Z-buffer. Each disk is assigned a different color. Based on the projection in four different directions from the viewpoint we can calculate the disks which are visible by looking at which pixels that are painted on the 4 planes. But this is also inaccurate as disks far away might be shown as invisible even though they are actually visible. Our algorithm is based on the notion that visibility changes at the tangents to the disk from the viewpoint. In this way, we can detect next visible disk within finite number of rays depending upon the total number of disks.

Section I: Computing the exact visibility

To compute the exact visibility of Disks we shoot two tangents rays D_iRay_{left} and D_iRay_{right} to every disk D_i from the viewpoint. (Here left and right are used for ease of notation and it does not matter if we switch between the two). If the corresponding ray (D_iRay_{left} or D_iRay_{right}) for the disk D_i hits the disk D_i without intersecting any other disk we mark disk Di as visible. In other words disk Di is closest disk from the view point along the ray D_iRay_{left}(or D_iRay_{right}). Furthermore we extend the ray (D_iRay_{left} or D_iRay_{right} or both) (along which Di is the closest from viewpoint) and find the next closest disk D_j along that ray. We mark D_j also visibile. Thus we require a total of 2*(number of Disks) rays to compute the exact visibility.

The pseudo code is given below

```
for (int i=0i<ndisks;i++)
   DiskVisibile[i]=false;
for (int i=0i<ndisks;i++)
{
   Rayleft= leftTangentToDisk(Viewpoint,Disk[i]);
  if(Disk[i] is closest along Rayleft)
  {
     DiskVisibile[i]=true;
    int j=-1;
  j=GetIndexofNextClosestDiskAlongRay(Rayleft);
if(j!=-1)
DiskVisibile[j]=true;
  }
   RayRight= RightTangentToDisk(Viewpoint,Disk[i]);
  if(Disk[i] is closest along RayRight)
  {
     DiskVisibile[i]=true;
    int j=-1;
  j=GetIndexofNextClosestDiskAlongRay(RayRight);
  if(j!=-1)
  DiskVisibile[j]=true;
  }
}
```

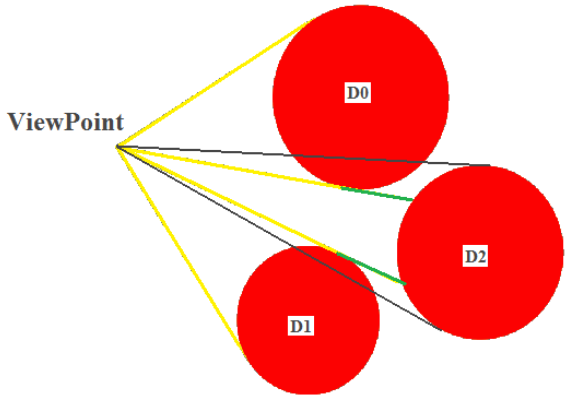


Figure 1: Yellow rays are tangent rays which mark D0 and D1 as visible. Green rays are extension of yellow rays which mark D2 as visible. It may happen that Green rays do not mark any disks visible. The black rays do not mark any disks as visible.

Proof: The above approach uses the fact that visibility changes at the tangent rays from the viewpoint to the disk. See figure for visualization of different cases.

Case I: The intersection of the tangent to the disk from the viewpoint and the disk is visible from the viewpoint. If the disk 'D' is closest from the view point along the tangent from viewpoint to 'D', then it is guaranteed that it is visible from the viewpoint. In figure 1, D0 and D1 are marked visible.

Case II: The disk is visible not at the tangent points. It can happen that some disks are visible even though the tangents for those disks from the viewpoint are cutoff by other disks. See figure 'b'. To take care of that we extend the rays for which we marked the disk as visible in step1 and mark the next disk which is closest from the viewpoint along the same ray. In figure 1, Disk D2 is marked because the yellow rays are extended (the green extension) which then intersect with D2.

Since the visibility changes at the tangents to the disks from the viewpoint and we check all such tangents we are guaranteed to find the exact visibility of all disks.

Complexity: For each disk we consider two rays. For calculating the first and second closest disk along each ray we need to touch all disks. Thus for approach is quadratic in nature ($O(n^2)$) where n is the number of disks. For the naïve approach the complexity is Rxn where R is the number of rays and n is the number of disk.

Section II: Comparison with Naïve approach.

To compare the two methods, we measured two factors required for exact visibility computation – execution time and number of rays required to find the exact visibility. Since the naïve solution is not guaranteed to always detect every visible disk, we defined a disk as "missed" if naïve solution marked it "not visible" when our solution marked it "visible".

When there are missed disks, we doubled the number of rays if it misses any visible disks. For example, if it misses any disk when it shoots 360 rays – one ray per one angle – we double it to shoot 720 rays, which is two rays per one angle. We tested the two methods with number of disks ranging from 10 to 1000(See Table 1). In all the cases, our method was significantly faster and required less rays to compute the visibility.

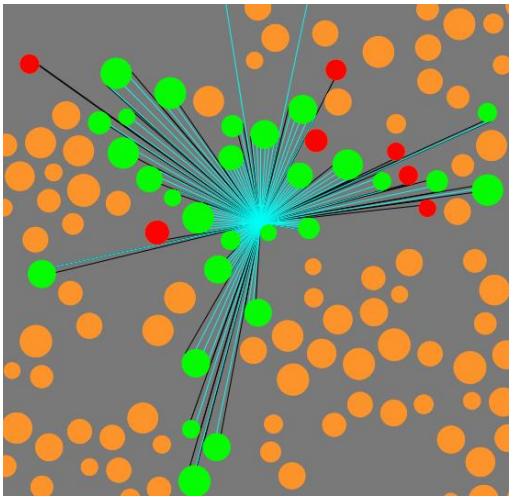
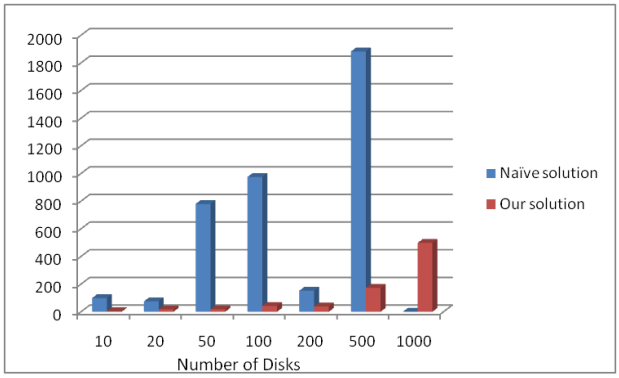


Figure 2: The disks marked as red were missed by the naïve approach. Our approach was able to detect them successfully. The green discs were marked as visible by both approaches.

Section III: Results

Table1: Computation time (in milliseconds)

# of disks	Naïve solution	Our solution
10	99	5
20	75	17
50	780	18
100	975	42
200	152	38
500	1883	173
1000	not applicable	498



Graph 1: The graph shows the execution time of the naïve approach and our approach

Table2: Number of Ray used by the algorithm.

# of Disks	Naïve solution	Our solution
10	20627	20
20	20627	40
50	330023	100
100	330023	200
200	82505	400
500	660047	1000
1000	Infinite	2000

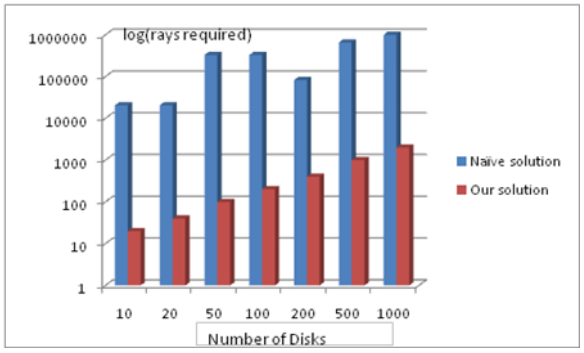


Figure 3: The Graph shows the number of rays used by the algorithm to determine exact visibility. Since the naive approach uses large number of rays we were forced to use log scale for this graph.

Section IV: Conclusion & Future Work

Based on the results presented in this paper we can conclude the following:

- The naïve approach may run faster when there are a lot of disks surrounding the viewpoint. In this case if there many disks our approach will shoot rays to all the disks, but the naïve solution may shoot very few rays and still find the exact solution. Example in Table1 , for n=200 we see that the performance of the naïve approach actually did improve when n was increased from 100. Though our approach still performed better even in this case for that particular disk configuration, it might happen than the naïve approach actually performs better as the number of disks increase further.
- Finding the number of rays required to determine the exact solution in naïve approach is difficult. We can only estimate it roughly by running the algorithm for a few configurations of the disks for a particular number of disks. But since the location of the disk can be totally random it is almost impossible to guarantee a value for the minimum number of rays required for the exact solution. But in our approach we always find the exact solution by shooting finite number of rays which is equal to twice the number of disks.
- In all cases (Table 1 and 2) we find that our approach performs faster than the naïve approach. There might be special cases where the naïve solution might perform faster as mentioned earlier in the conclusion. But based on the results our solution is deterministic and works better in general.

Future Work

We have one drawback in our approach. If a ray is tangent to more than one disk we might extend it and mark a disk as visible incorrectly. We could add a check for this and not extend that ray to avoid such scenarios. Though our approach performs better than the naïve approach there is scope for speedup. Our approach touches all the disks. We could use concepts like BSP trees and occlusion principles

to avoid processing all the disks and just calculate the visibility based on limited candidate disks. This might significantly improve the performance of our algorithm.

References

[1] J.Rossignac, CS7491 Project 2: Visibility. http://www.gvu.gatech.edu/~jarek/courses/7491/7491P2-2008.pdf

[2] J.Rossignac, Visibility and Occludeers, http://www.gvu.gatech.edu/~jarek/GVC/slides/shodows2008.ppt

[3] P.Bourke, Intersection of a Line and a Circle. http://local.wasp.uwa.edu.au/~pbourke/geometry/sphereline/

[4]Wikipedia, Tangents to a Circle, http://gpwiki.org/index.php/Tangents_To_Circles_And_Ellipses#Tangents_to_a_Circle