Good Looking Circles.

Kazimir Majorinc.

We attempt to draw a simple illustration for a thermodynamics textbook. The molecules in fluid are represented with circles of equal size. The circles can touch, but not intersect each other. They should be densely positioned, i.e. it should be impossible to draw an additional circle without intersecting others. The whole picture should look *natural*. That condition is vague but unavoidable: the picture should not look like it is drawn by human, or by any other, unrelated algorithm. Also, the program does not need to be fast; only one good picture need to be drawn for the textbook.

The problem is solved in four attempts; all four are described in this article. A few elements are common to all attempts. The record *circle* and procedure *draw_circle* are defined; they only make the program slightly shorter and simpler. The distance between two circles, d, is defined as in geometry; actually, even bit more generally: it is assumed that the distance between intersecting circles is negative.

```
link random
link graphics
global SIZE, R
record circle(x, y, r)


procedure d(C1, C2)
  return sqrt((C1.x-C2.x) ^ 2+(C1.y-C2.y) ^ 2)-C1.r-C2.r
  end

procedure draw_circle(C)
  DrawCircle(C.x, C.y, C.r)
  return C
  end

procedure eliminate_intersecting(Candidates, c)
  every insert(ToDelete:=set(), d(c, c0:=!Candidates)<0 & c0 )
  return Candidates --:=ToDelete
  end


procedure main(args)
  SIZE:=150
  R:=10
  every (SIZE|R) := get(args)
  randomize()
  attempt1()
  attempt2()
  attempt3()
  attempt4()
  write("Done")
  WDone()
  end
```

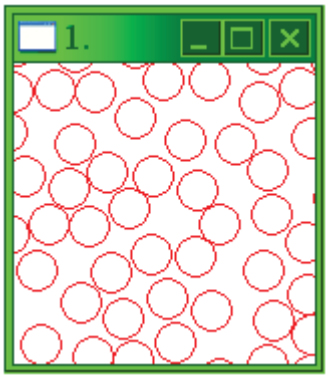
As the running time is not important, all algorithms are left in the simplest, unoptimized form. After a circle is drawn, in the next step, we form the set *Candidates* that contains all circles that satisfy criteria for a next step. Most importantly, they do not intersect any already drawn circles. One of the candidates is randomly chosen and drawn. This process repeats until *Candidates* is empty. In three of four attempts, the *Candidates* set initially contains all circles that could be drawn in quadrant of the given size.

The First Attempt.

In the first attempt no additional criteria are imposed. The procedure that draws a random set of the non-intersecting circles is short and simple.

```
procedure attempt1()
   &window:=WOpen("height="||SIZE, "width="||SIZE, "label=1.", "fg=red")
   every insert(Candidates:=set(), circle(-2*R to SIZE+2*R, -2*R to SIZE+2*R, R))
   while Candidates:=eliminate_intersecting(Candidates, draw_circle(?Candidates))
   end
```

Some circles are drawn outside of the visible part of the picture, to avoid possible edge effects on the picture.



This picture is not satisfactory: the circles are not dense enough and the parts left uncovered by circles are too big. As all circles are chosen randomly, the program can, theoretically, draw denser picture; however, it is not likely to happen in a reasonable number of executions.

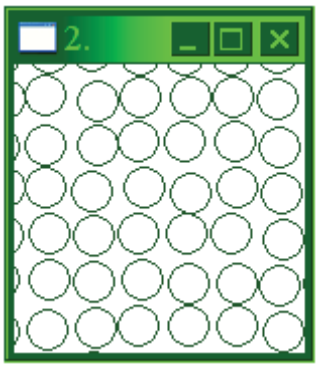
The Second Attempt.

It was suggested that the program could choose circles that form a perfect quadratic or hexagonal pattern, and then slightly translate individual circles in the random direction. Ideally, both higher density and randomness can be achieved.

```
procedure attempt2()
   distance_between_centers:=(2*R+3)
   tolerance:=3
   &window:=WOpen("height="||SIZE,"width="||SIZE,"label=2.","fg= dark green")
   every insert ( Candidates:=set(),
                  i:=-2*R to SIZE+2*R &
                  j:=-2*R to SIZE+2*R &
                  (i+100) % distance_between_centers <= tolerance &
                  (j+100) % distance_between_centers <= tolerance &
                  circle(i,j,R)
                )
   while Candidates:=eliminate_intersecting(Candidates, draw_circle(?Candidates))
   end
```

The algorithm is similar to the previous one. Additionally, an initial choice of the candidates is narrowed to the circles with coordinates of the centres in the segments $[k(2R+3), k(2R+3)+3]$, $k=0, 1, 2 \ldots$ We added the constant 100 to avoid aperiodicity of the operation % around zero.

However, the result of the second attempt is even less satisfactory.



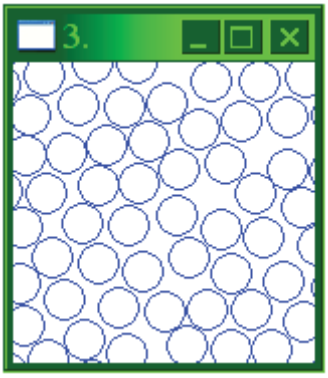
Although the circles are dense, their distribution is too regular. The square pattern is clearly visible, especially if seen from a distance. Variations of the values *tolerance* and *distance_between_centers* do not help: the pattern can become less obvious only if the density is significantly decreased.

The Third Attempt.

In the third attempt we applied a similar idea, but more locally. If circles are chosen so they touch at least two already drawn circles, then the result is a perfect hexagonal pattern. We expected that, using a slightly looser requirement: circles are drawn so they are close to two other circles, density can be preserved, while irregularities will accumulate to the degree that hexagonal pattern will not be clearly visible in picture.

```
procedure attempt3()
   min_allowed_d:=0
   max_allowed_d:=5
   &window:=WOpen("height="||SIZE, "width="||SIZE, "label=3.", "fg=blue")
   every insert(Candidates:=set(), circle(-2*R to SIZE+2*R, -2*R to SIZE+2*R, R))
   Drawn:=[ ]
   repeat
   { Candidates2:=[ ]
      every c:=!Candidates do
      { close_circles:=0
         every (min_allowed_d <= d(!Drawn, c) <= max_allowed_d)
         do close_circles+:=1
         if (close_circles > 1) | (close_circles = *Drawn <= 1)
         then put(Candidates2, c)
      }
      if not(c:=?Candidates2) then fail
      Candidates:=eliminate_intersecting(Candidates, draw_circle(c))
      put(Drawn, c)
   }
   end
```

Our expectations are fulfilled and resulting picture looks significantly better; it is both dense and irregular.

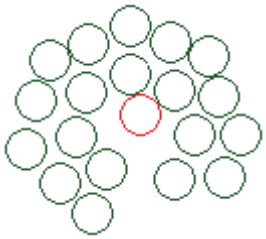


Some circles form hexagonal pattern, but it looks more like the natural tendency of the densely packed circles than the result of some inadequate algorithm.

However, as can be seen from the picture, some relatively large empty areas occurred again on pictures drawn with *attempt3*. Although it is not obvious from the above static picture, observation of the program during work clearly reveals the origin of such empty spaces.

The set of the all drawn circles can form figures with large concavities that are bigger than one circle. Some of these concavities can not be filled with densely packed circles.

See, for example, the picture below: the green circles form a concave figure, and after red circle (one in the center if you see this article in monochrome) is drawn, the large area around it can not be filled any more.



The Fourth Attempt.

The problem with the previous attempt can be solved easily. We define the center of the first drawn circle as the *Center* of the whole picture. Beside the previously mentioned criteria, drawn circles are chosen to be relatively close (variable *tolerance*) to the *Center*. If *tolerance* is 0, the circle closest to the center that does not intersect with other circles is drawn. This ensures that figures formed by the circles have no large concavities.

```
procedure attempt4()
   min_allowed_d:=0
   max_allowed_d:=5
   tolerance:=5
   &window:=WOpen("height="||SIZE, "width="||SIZE, "label=4.", "fg=black")
   every insert(Candidates:=set(), circle(-2*R to SIZE+2*R, -2*R to SIZE+2*R, R))
   Drawn:=[ ]
   repeat
   { if /Center
        then c:=(Center:=?Candidates)
        else { Candidates2:=[ ]
               min_d:=1E5
               every c:=!Candidates do
               { close_circles := 0
                 every (min_allowed_d <= d(!Drawn, c) <= max_allowed_d)
                    do close_circles+:=1
                 if (close_circles > 1) | (close_circles = *Drawn <= 1)
                    then { put(Candidates2, c)
                           min_d >:= d(Center, c)
                         }
               }
               every put(Candidates3:=[], (d(Center, c:=!Candidates2)<=min_d+tolerance) & c)
               if not(c:=?Candidates3) then fail
             }
     put(Drawn, c)
     Candidates:=eliminate_intersecting(Candidates, draw_circle(c))
   }
   end
```

Finally, the result satisfied all our criteria; at least sufficiently well for the textbook illustration.