CS 6491 - Project 1 - Packing Game

Anshul Bhatnagar
anshul.bhatnagar
@gatech.edu

Gaurav Dhage
gr8dhage
@gmail.com

Chris Martin
chris.martin
@gatech.edu

Suraj Sirpilli
surajsirpilli
@gatech.edu

Abstract

We implemented a two-player game in which players arrange a collection of non-overlapping disks on a plane. The objective is to minimize the size of the smallest circle that encloses all of the disks. The field of play is set up with two identical sets of disks so game can be played between two humans or against the computer.

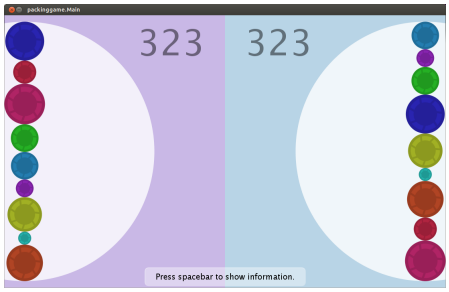

Figure 1: A game's starting configuration.

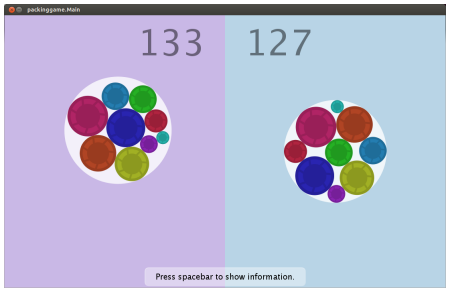

Figure 2: A human's best attempt (left) and our packing algorithm (right).

1 Collision detection

The game rules disallow overlapping disks, and we want to make an interface that allows a player to easily position disks directly adjacent to one another. So when a mouse drag attempts to move a disk to a location that would cause overlap, we instead move the disk to the nearest valid location, as illustrated by Figure 3.

When a disk is being moved, we model the situation as a point being moved among disks that are expanded by the radius of the moving disk. If the point is inside any of the expanded disks, this indicates overlap and therefore an invalid position.

There are two collision resolving methods, both of which are depicted in Figure 4:

1. If the point intersects a single expanded disk, then we first try to position the moving disk at the edge of the disk that overlaps.
2. If the first method that position overlaps some other disk, then the closest position for the point is at an intersection of two expanded disks. We simply choose the closest among all of the $O(n^2)$ intersection points.

2 Minimum enclosing circle

Given a fixed center point c, we calculate the minimum enclosing radius as

$$\max(\text{radius}(d) + \text{dist}(c, \text{center}(d)))$$

over all disks d.

The nontrivial part of the minimum enclosing circle problem is to find the optimum center point. Our solution is a close approximation based on a random walk with a progressively decreasing step size. We

Figure 3: The blue disk at the bottom is being dragged upward. A thin black outline around the mouse cursor indicates where the disk would be if it were not blocked by the others.

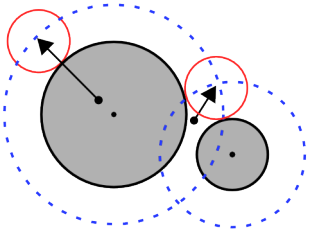

Figure 4: Two disks (gray), their expanded boundaries (dashed blue), and two examples of shifts from overlapping positions to their closest valid positions.

repeatedly generate a random vector and use it to shift the center if the shift results in an improvement to the minimum radius.

3 Disk Packing

Our general approach is simply to iterate through the list of disks, placing each disk into a valid position based on some greedy heuristic. We considered an *outward* strategy of placing each disk in the position that results in the minimum increase in the enclosing circle's radius, but experimentation pointed us instead to a more effective *inward* strategy of starting at the perimeter of some enclosing circle and working in toward the center.

The inward strategy requires knowing the radius of the enclosing circle before placing any disks, and it may fail if the radius is too small. So we do many packing attempts with different circle sizes, using a binary search to find the minimum radius such that the packing algorithm succeeds.

We found no reliable best way to choose the order

$r_{\min} \leftarrow \sqrt{\text{sum of disk areas}/\pi}$
$r_{\max} \leftarrow$ sum of disk radii
$P \leftarrow \perp$
while $r_{\max} - r_{\min} > \varepsilon$
  $r \leftarrow \text{average}(r_{\min}, r_{\max})$
  $p \leftarrow$ pack (Algorithm 2) with radius r
  if packing was successful
    $r_{\max} \leftarrow r$
    $P \leftarrow p$
  else
    $r_{\min} \leftarrow r$
return P

Algorithm 1: Radius selection

in which disks are placed, so we simply try various randomly-selected permutations.

We search for candidate disk positions using the same point and expanded-disk model used to deal with multi-disk collisions, but in this case we also must respect the constraint imposed by the enclosing circle. This still easily fits within the model, however – we simply need to *reduce* the enclosing circle's radius rather than expand it. We the position each disk by choosing the valid intersection point farthest from the center of the enclosure.

---

repeat 100 times
  randomly permute list of disks
  position first disk arbitrarily at enclosing edge
  for disk d in remaining disks
    $B \leftarrow$ [positioned disks expanded by d.radius)]
      + [enclosing circle reduced by d.radius]
    $I \leftarrow \{i \in \cup_{(b_1,b_2)\in B^2}\ b_1 \cap b_2 : i \text{ is valid}\}$
    if $I = \phi$ then continue
    d.center $\leftarrow \arg\max_{i\in I}$ [distance$(i,$ center$)$]
  return disk positions
return $\perp$

Algorithm 2: Disk packing

4 References

Bourke, Paul. *Intersection of two circles.* `http://paulbourke.net/geometry/2circle/`