Recitation 2.3

April 7, 2025

Outline

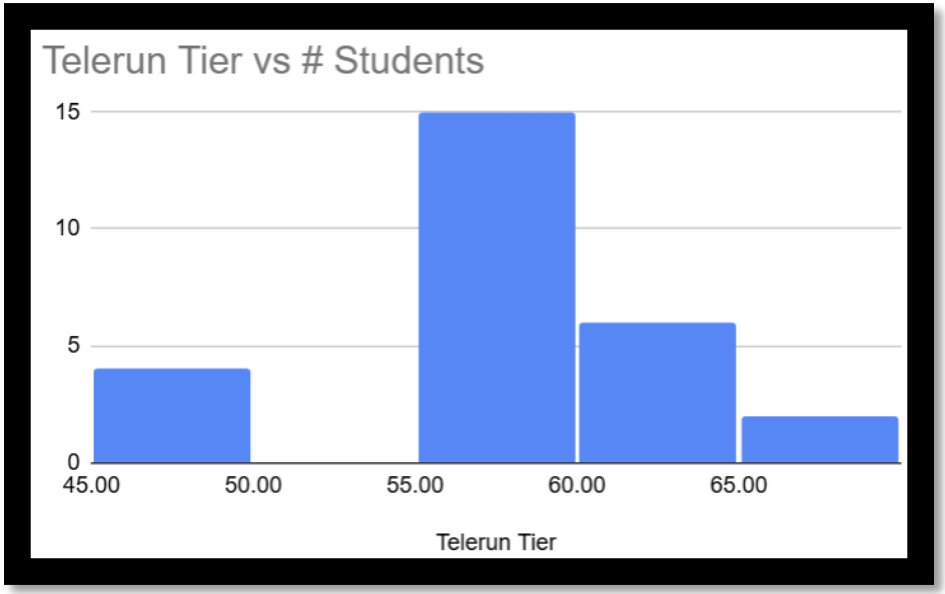
1. Q&A on HW5
2. HW6 Checkoff announcement
3. Project Beta Tiers and Tiers for Final
4. Project 2 Final Push:
5. Sweep and Prune
6. Next steps

HW6 Checkoff

- Will be released on Tuesday (4/8)
- Encourageed due date is Sunday (4/13) (can be submitted as late as next Friday (4/18) regardless)
- **Task:** Complete HW6 check-off and brush up on Lectures 14 and 15.
- HW6 has no Git repository. It is a theoretical write-up!
- An overview will be covered in Recitation 2.4.

Project 2 Beta Tiers

- Project 2 Beta full point tier : 50
- (Yes, you have all surpassed the beta expectations!)
- Project 2 Final Guarantee B grade Tier : 60
- (No, your performance in beta did not influence this)
- Project 2 Bonus points cut off tier – Will be announced next week



Project 2 remaining optimizations

- Improved collision detection (Today's focus)
- Improve Render using your knowledge from project 1 and HW1-HW4.
    - Highly recommended: **Vectorization**
- Be guided by perf reports for further optimizations.
- Following every recitation for project 2 so far might not be enough for the bonus - Improve on top of what you have been told to do, to reach the tier necessary to get the bonus.

Base Collision Check

```
for (int i = 0; i < state->s_spec.n_spheres; i++) {
  for (int j = i + 1; j < state->s_spec.n_spheres; j++) {
    if (check_for_collision(state->spheres, i, j, &minCollisionTime)) {
      indexCollider1 = i;
      indexCollider2 = j;
    }}}
```

Each check_for_collision call involves computation :

- Euclidean Distance between two spheres (has sqrt)
- Relative velocity of two spheres
- Use relative velocity to find if collision will happen (another sqrt)

Expensive!

One improvement

For each sphere,

- Find the ending position of the sphere for the timestep of the simulation
- Use its starting and ending position, to have a bounding box for the sphere for potential collisions

Before collision checks,

- Check if the box can collide with another box by comparing the individual coordinates of the boxes (no square roots and only uses 3 comparisons)
- The above filter provides false positives for collisions but no false negatives – Double check with the original check_for_collision on false positives.

How to further improve without losing correctness?

- Notice how the comparisons in the previous technique is such that if checks on even 1 dimension fails, then collision cannot happen.
- Hence, we can reduce the initial “filter” to 1-dimension checks and check the other dimensions as and when necessary.

Today's focus: The Sweep and Prune algorithm

- A good reference: Sort, sweep, and prune: Collision detection algorithms
- The algorithm has the following three parts:
- Sort : Sort the box coordinates based on 1 dimension (say X axis)
- Sweep : Filter out impossible collisions by inexpensive box comparisons
- Prune: Double-check to ensure correctness (by removing false positives)

The Sort

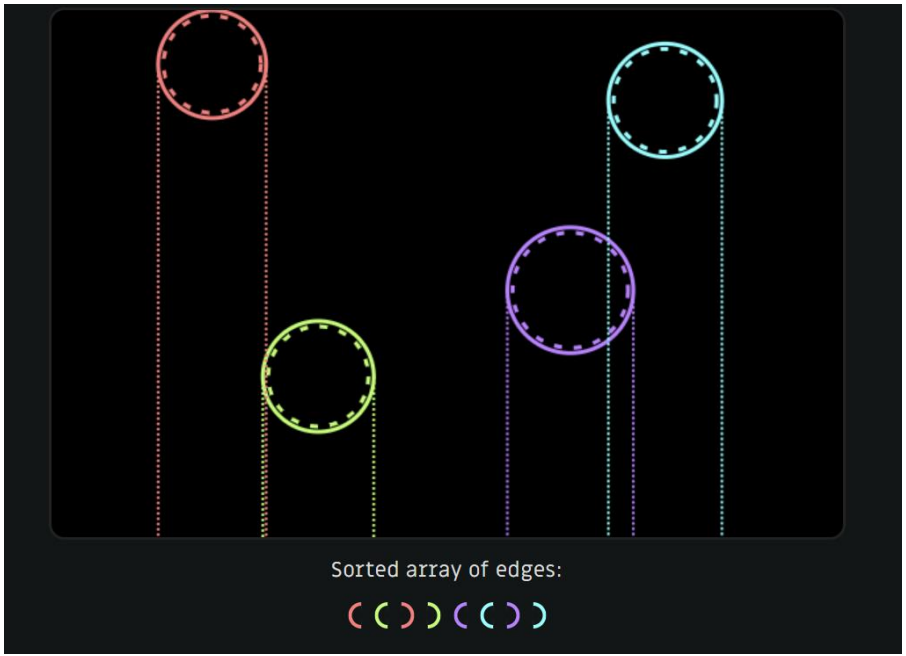


Image from : https://leanrada.com/notes/sweep-and-prune/

- Sort the spheres using a box that entails the spheres possible locations within the timestep.
- (Figure shows a sphere as image was taken from the reference, but think of them as boxes for our project)

The Sweep

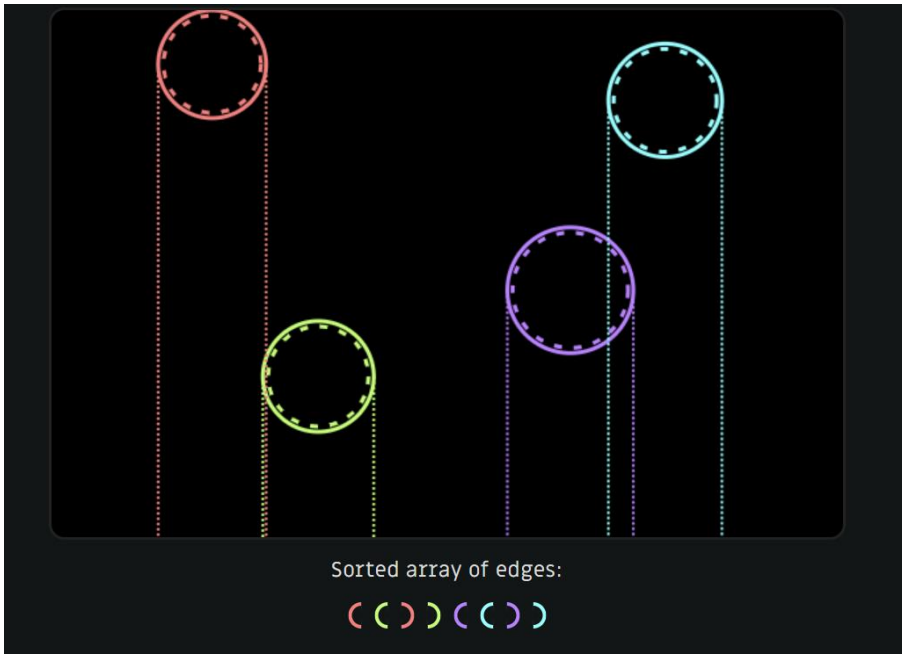


Image from : https://leanrada.com/notes/sweep-and-prune/

- For each sphere, eliminate collision checks with coordinates further away from the last check that cannot collide.
- Example: Red box cannot collide with purple, hence we don’t have to check collision with blue)

The Prune

- Now that 1 dimension has filtered out a lot of checks,
- Check for the other dimensions with the box comparison.
- Those that pass, maybe false positives (may not actually be colliding) – Use the default check only on these.

Why is it that it can be faster?

- The worst case maybe the same – Every box collides with every other box
- But when the balls are well scattered (happens to many pairs in this project), it reduces the number of checks involving square root operations significantly.
- I suggest reading the reference that beautifully explains the algorithm with better visualization (Don’t miss its second page!) : https://leanrada.com/notes/sweep-and-prune/

Next steps?

- Beyond this optimization, run perf reports to see which of the two (render, simulate) is worth focusing your optimizations on.
- If only the recitation optimizations have been performed, it is very likely to be render.
- Use perf reports to guide you on what could be improved.
- Vectorization can be very powerful in many steps involving the image computation.
- Aim to reach the bonus tier!

Attendance

Lab Session

- Questions?