Graphics Term Project
Sphere Packing

•••

Fred Barthel, Parker Carlson, Martin Soto,
Bryan Stoffel, Megan Waterworth

Roles

Sphere Packing Algorithm (Normal and Dense Packing)
Parker Carlson, Martin Soto, and Bryan Stoffel

OpenGL Graphics/Sphere Viewing/Animations
Fred Barthel and Megan Waterworth

Introduction

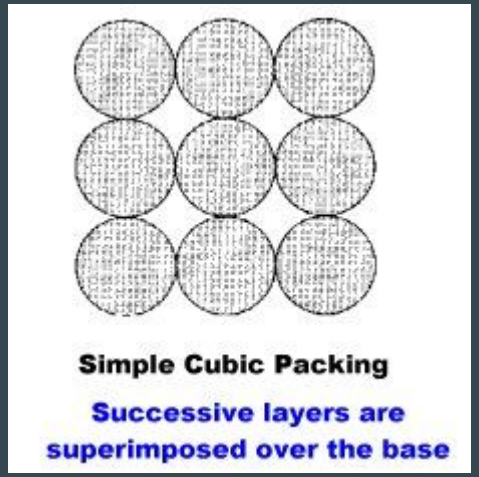
Problem Statement

Given a cubic volume and a sphere radius, determine how many spheres can fit in the cubic volume and visualize them.

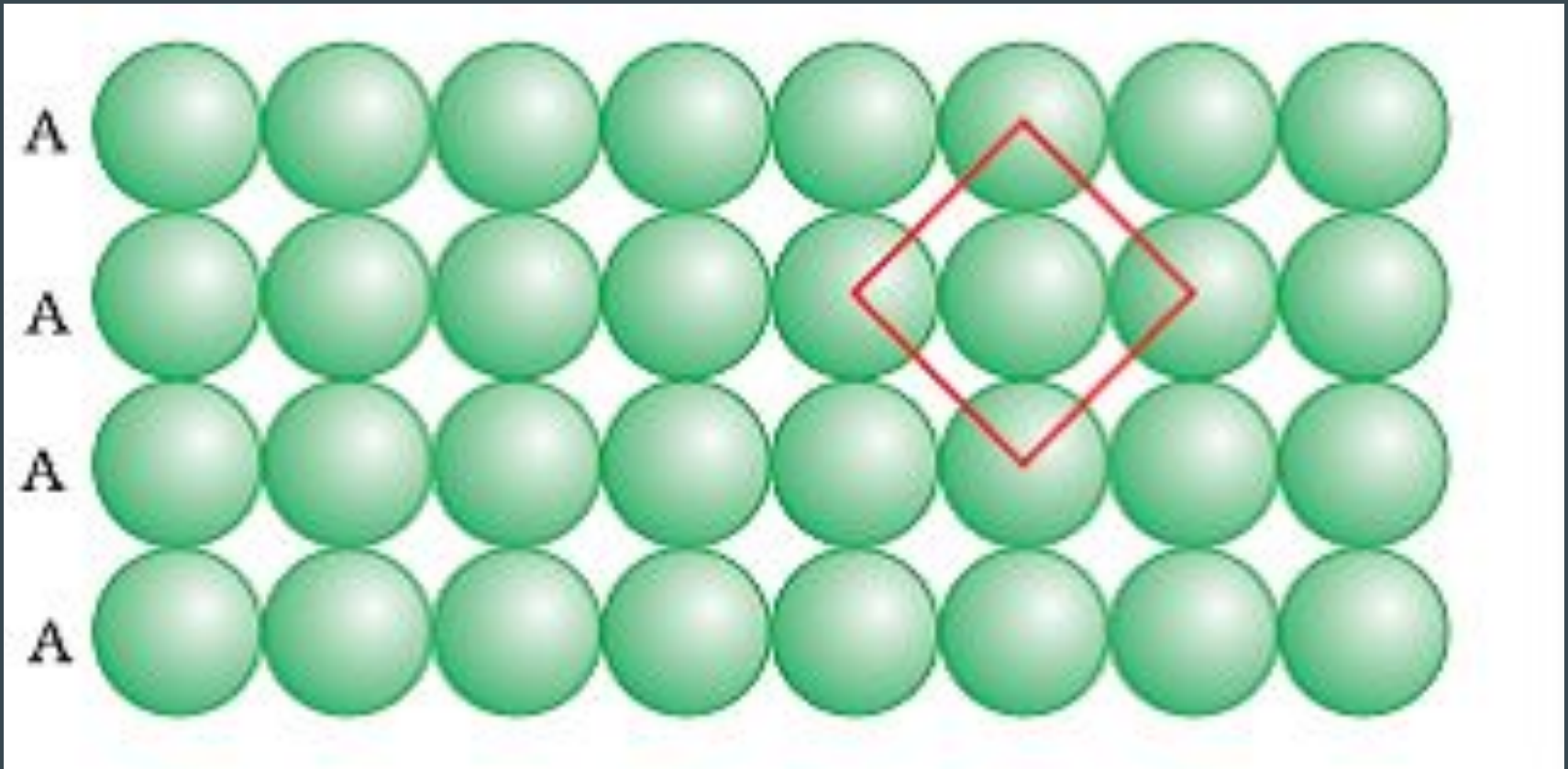
Method/Theory

Normal Packing

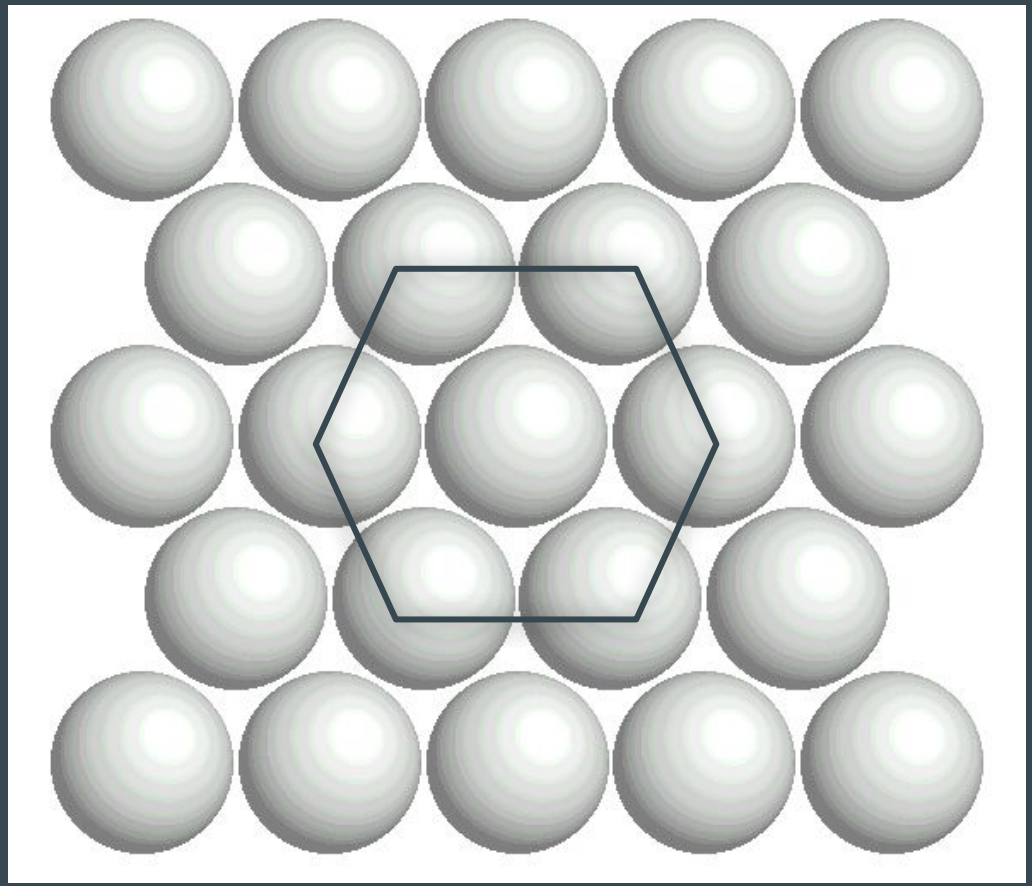
- **Packing Method:**
  - Cubic Crystalline Lattice Packing
- **Thought Processes:**
  - Visualize the most normal pack to maximize number of spheres
  - Place the spheres in straight up, down,left and right positions 2r away
  - Use 2D array to hold centers of each array and its radius
- **Equations for finding centers:**
  - Start with first center being at {r,r,r} away
  - Proceed to update by {2r,r,r} until far wall is reached (length)
  - Then update by {r,2r,r} once to shift row back each time the wall is reached (width)
  - Finally once the current layer is filled, update by {r,r,2r} to move the row upwards and begin to find centers at the new level (height)



Sphere Normal Packing



Sphere Dense Packing

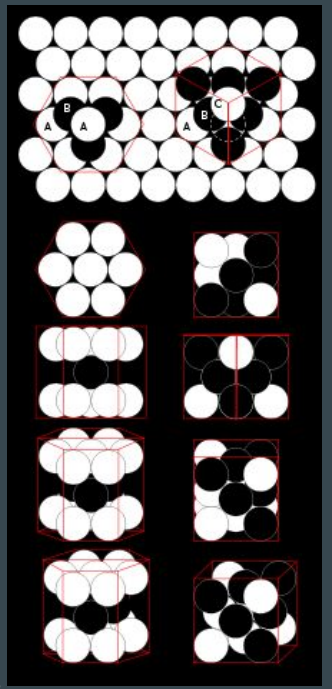


Dense Packing - Initial Thoughts/Research

- Thought Process:
  - Initially we needed to determine how many spheres can fit in the box if they are packed as densely as possible using Hexagonal Close Packing (HCP).
  - Once we find out the number of spheres we can then place them into the box and know when to stop adding spheres based on how many will fit.
- Equation:
  - (Volume of Box/Volume of a sphere) * Average Density = Number of Spheres
  - Average Density = $\pi/(3 * \sqrt{2}) \backsimeq 0.74048$

Dense Packing - Finalized Thoughts/Research

- Packing Method:
  - Hexagonal Close Packing
- Thought Process:
  - We needed to then find out the distance each sphere center is from the sides of the box in terms of their radius (r).
  - We then place the center into a 2D array based on layer and center coordinates.
- Equation for each Sphere Center:
  - Even Layers:
    - Even Rows: (2r, r, r), (4r, r, r), (6r, r, r) , . . .
    - Odd Rows: (r, r + √3r , r), (3r , r + √3r, r), (5r, r + √3r, r), . . .
  - Odd Layer:
    - Even Rows: (r, r + (√3r)/3, r + (√6r2)/3) , (3r, r + (√3r)/3, r + (√6r2)/3), . . .
    - Odd Rows: (2r, r + (4√3r)/3, r + (√6r2)/3) , (4r, r + (4√3r)/3, r + (√6r2)/3), . . .



OpenGL Viewing

The spheres are stored in two 2D arrays which are:

[*numNormalPack*][4] & [*numDensePack*][4]

We render the spheres first (*drawSpheres()*), then the walls (*drawWalls()*).

In *drawSpheres()*, we used the following functions from the OpenGL library, on each sphere in the array.

- *glutSolidSphere(radius, slices, stacks)*
- *glTranslatef(x, y, z)*

In *drawWalls()*, we used the following functions from the OpenGL library.

- *glBegin(GL_POLYGON)* and *glEnd()*
- *glVertex3f(x, y, z)*

Transparency Technique

- The walls are drawn transparently so the spheres can be seen more clearly.
- Multi-layer transparency in OpenGL requires careful consideration of render order
- Opaque surfaces are rendered first
- For the most realistic appearance, overlapping transparent surfaces should be rendered from front to back
    - setWallOrder() uses camera position to determine render order for the walls and ceiling of the box

OpenGL Animations

When the size of the box changes in *keyboard()*, we:

1. Call *translateSpheres()* to determine the new number of spheres that can fit in the new box.
   a. Calls *getCoodsNormalPack()* and *getCoordsDensePack()*
2. Call *glutPostRedisplay()*
   a. Calls *drawSpheres()* and *drawWalls()*

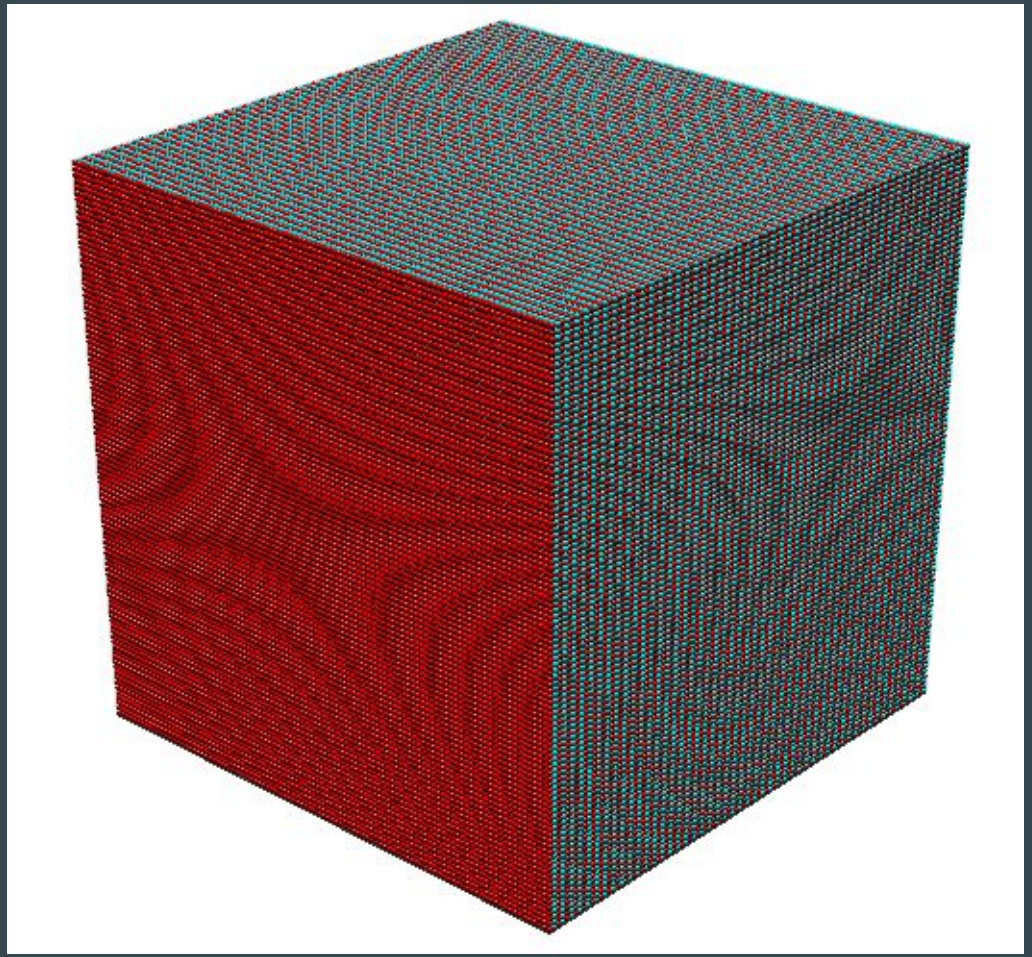
We also have included an option to switch between Normal and Dense Packing by switching `packingMode` and re-rendering.

We used Double Rendering (*GLUT_DOUBLE*) to make the animations smoother.

Results

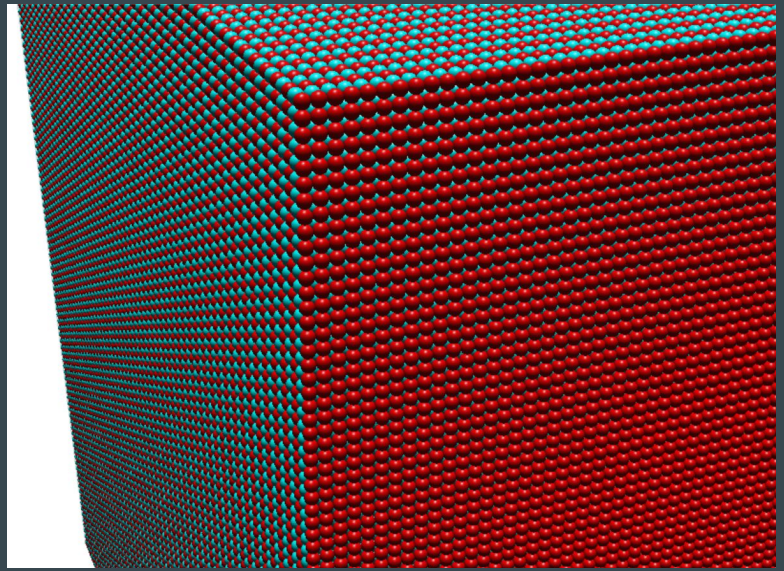
Normal Packing

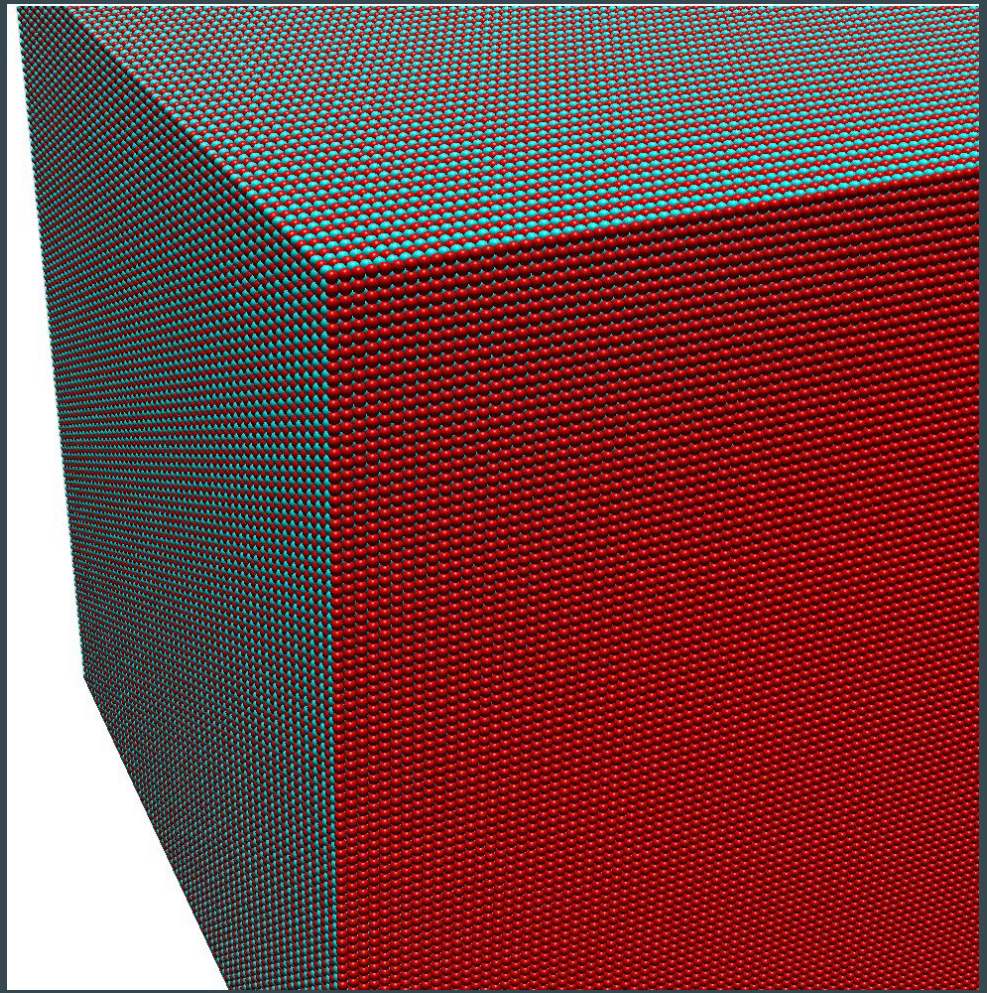
- 200 x 200 x 200 box
- Sphere Radius = 1
- 1 million spheres



Normal Packing

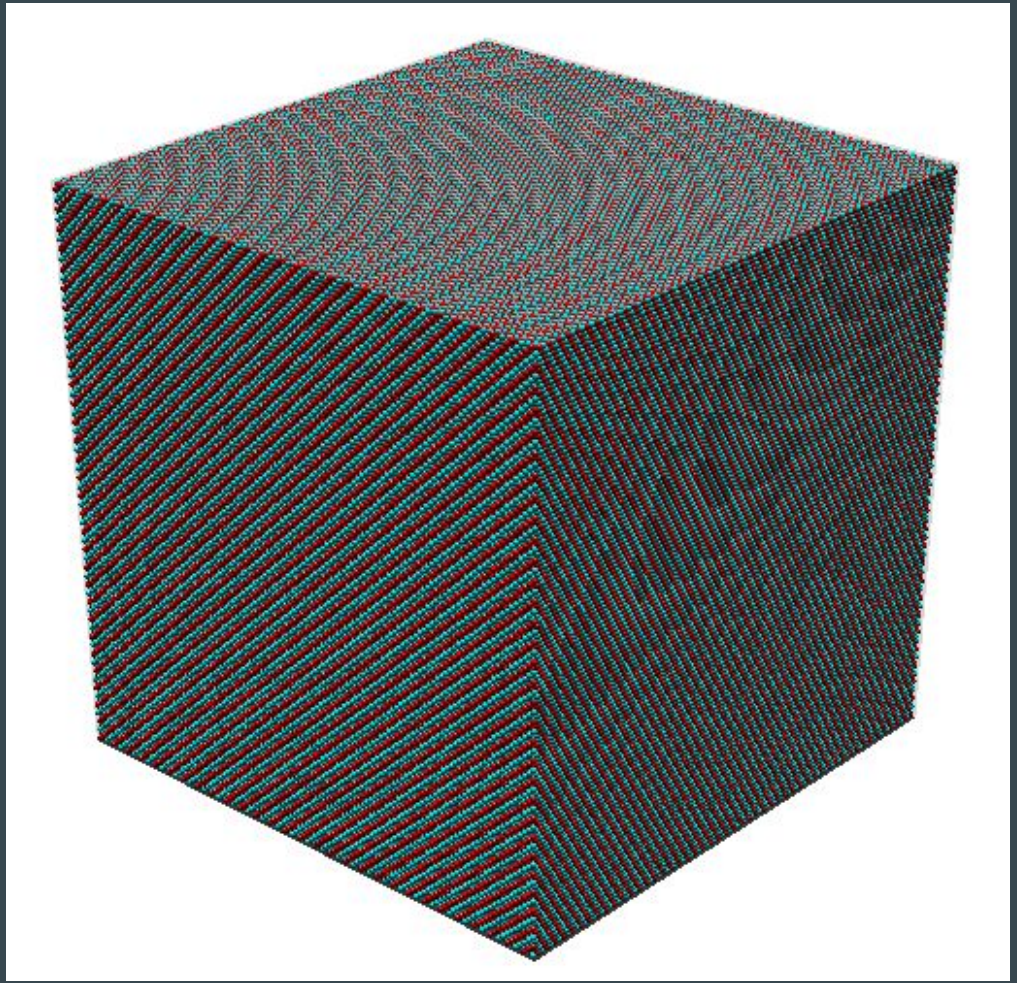
- 200 x 200 x 200 box
- Sphere Radius = 1
- 1 million spheres





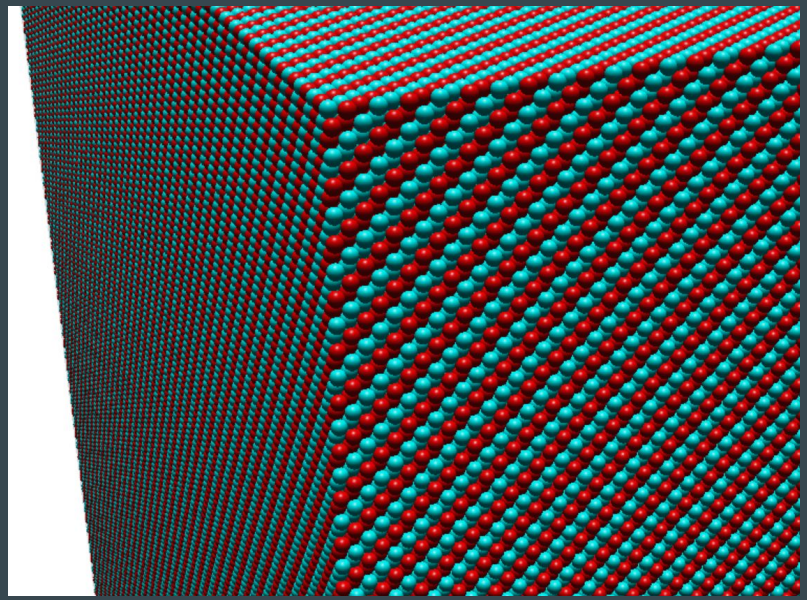
Dense Packing

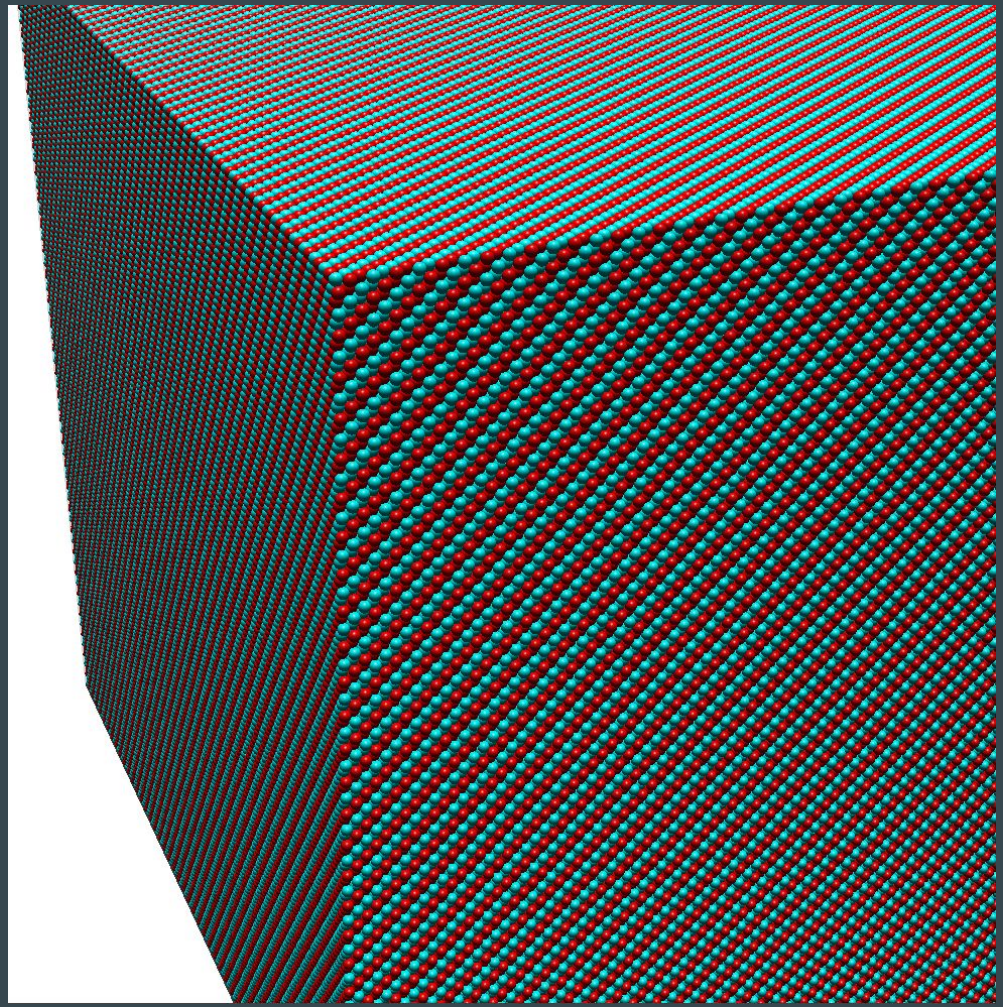
- 200 x 200 x 200 box
- Sphere Radius = 1
- 1,395,985 spheres



Dense Packing

- 200 x 200 x 200 box
- Sphere Radius = 1
- 1,395,985 spheres





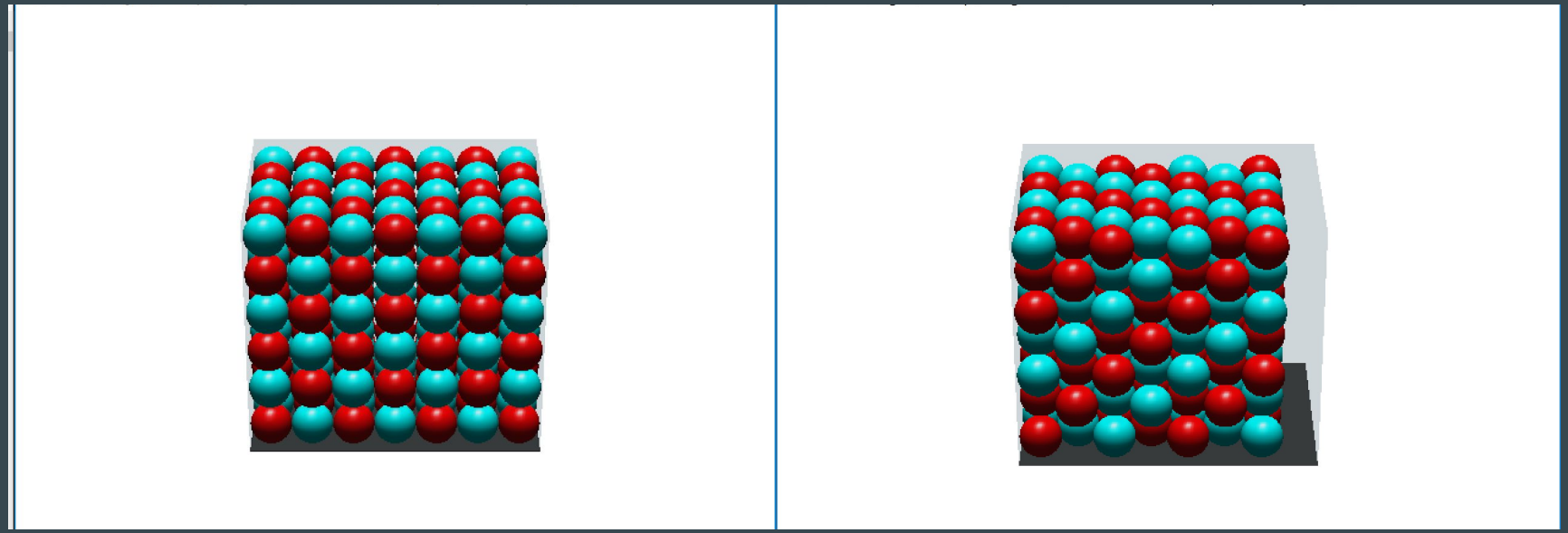
OpenGL Viewing Controls

```
Controls:
                    x/X ---------------------- decrease/increase box length
                    y/Y ---------------------- decrease/increase box height
                    z/Z ---------------------- decrease/increase box depth
                    -/+ ---------------------- zoom out/in
                    p ------------------------ switch pack normal/dense
                    i------------------------- reprint instructions
                    ESC ---------------------- close
                    Arrow Keys ---------------- move camera view
```

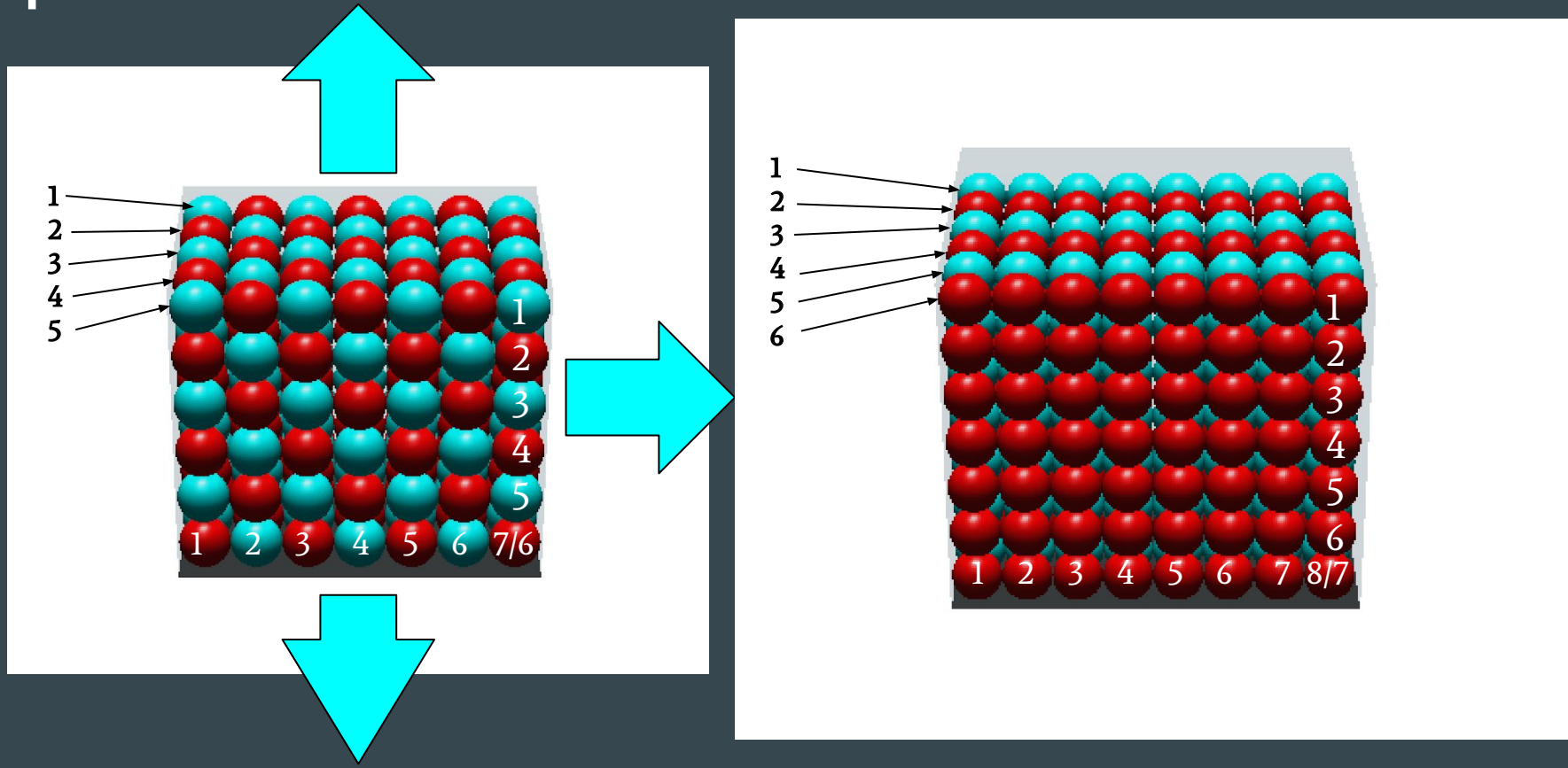
OpenGL Viewing

```
Enter the length of the Box in Inches: 10
Enter the width of the Box in Inches: 14
Enter the height of the Box in Inches: 12


Enter the radius of the Spheres in Inches: 1
Normal packing real yields: 210 Spheres
Dense packing real yields: 221 Spheres
Dense Pack has 11 more.
You should use Dense Pack!
```



OpenGL Animations



Conclusion

Final Thoughts

Overall, we were able to complete:

- Normal sphere packing algorithm, returning the number of spheres that can fit and their centers
- Hexagonal dense sphere packing algorithm, returning the number of spheres that can fit and their centers
- Visualization of the spheres when using both normal and dense packing
- Animation of expanding box and re-calculation of packing
- Switch visualization between dense and normal packing

References

- Close Sphere Packing - https://en.wikipedia.org/wiki/Close-packing_of_equal_spheres
- GLUT Functions - https://www.opengl.org/resources/libraries/glut/spec3/spec3.html
- GL Functions - https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/

Demo