Packing of particles

Shruti Jain
MSc Computer Animation and Visual Effects
Bournemouth University

24th Aug 2015

Abstract

Particle packing has been a topic of interest in many areas like engineering, mathematics, physics, biology and computer graphics. Though this topic is not much explored in the field of computer graphics, there are many algorithms proposed in other fields. Many papers can be found in the field of powder technology. In the field of computer graphics, packing of particles can be really interesting in itself as a structural representation or as a packed object that can be used as the basis for different types of simulations. An object packed with particles can be used for fluid simulations or act as the input geometry for point based dynamics. Different types of packing have been implemented, broadly categorized as lattice and non-lattice. The nature of the lattice packing is repetitive and layers resemble other layers, depending on the type chosen. In non-lattice packing, random packing being the most interesting, has been implemented. The size of the particles is kept same for lattice packings. But, in case of random packing, the approach is not confined to the same size of the particle, hence particles can have different sizes. The different implementations are based on various technical papers, and some methods have been combined and even extended.

Contents

List of Figures

1 Introduction

Several algorithms have been proposed to create packed particles in different types of arrangements. The key factors to gauge the algorithm would be efficiency, complexity, adaptivity and outcome. Efficiency is an important factor to be considered for any algorithm, but based on the expected outcome of packing, efficiency may need to be compromised. For example, lattice packing usually will be faster but if the particles need to be packed randomly then the time taken to generate the packing will exceed. The term complexity refers to the complex nature of the algorithm, hence complexity is another factor that is related to efficiency. If the algorithm is complex it will take more computation time.

This report is based on generation of particles confined to a mesh. Different particle packing methods have been implemented to generate various forms of arrangements. The commonly known close-packing methods have been implemented. These methods are known to generate dense packings. One of the other implemented methods is generation of particles based on the Lattice vectors. This method is implemented for different shapes other than spheres, where platonic solids have been used as the choice for shape. The implemented lattice vectors are amongst the ones known to generate dense packings, and so they vary from one shape to another. There are multiple variations using the combination of particle shape and packing type, or lattice vectors, that can be possibly made. Therefore, some of the combinations have been implemented, but more emphasis has been given to random packing.

Random packing of particles is an important aspect, as most of the packings are random in nature. Also, efficiency of the algorithm and density of packing play a vital role. Considering the packing space as one entity, it will be exhaustive to achieve denser packings by merely generating random positions and checking for overlaps across the entire packing space. For faster computation, the mesh is divided into cells, and the cells in turn maintain the list of particles belonging to itself. So whenever a random position for a particle is generated, the particle is checked against the particles in its own cell and the surrounding cells. Since not all the particles present in the packing space needs to be checked, this reduces the computation time considerably. The packing resulting for this method will not be dense, unless there is a way to move the particle or change its radius, based on the neighbouring cells. These two options have been added as part of the two different approaches implemented for random packing.

2 Related work

Not many papers focus on packing of arbitrary shapes, but Jia and Williams [2001] proposed an approach that mentions about it. This approach was based on digitization in which the particle and the packing space both are digitized. The packing space is divided into 3D lattice and the particle behaves as a collection of voxels. The cells in which particles exists are marked, which makes the collision detection easier as it only means checking if the cell has a value or not. Simulation of non-spherical hard particles which is driven by collision was published by Donev et al. [2005]. The simulation was confined to a parallelepiped domain satisfying periodic or hard-wall boundary conditions. A detailed algorithm for making partial updates to near-neighbour list was presented. The concept is based on molecular dynamics simulations. Conway and Torquato [2006] suggest that the convex shape having the smallest packing density might be the regular tetrahedron. Other problems related to packing and tiling of tetrahedra were presented . In 2009, dense packing of Platonic and Archimedean solids was published [Torquato and Jiao, 2009]. They suggested that the densest packings of platonic shapes, other than tetrahedra, are Bravais lattice packings. The lattice vectors for the densest packing of the individual shapes have been provided. Also, the optimal densest packing lattice vectors have been provided for Archimedean solids, other than truncated tetrahedron which has denser non-lattice packings. The densest known optiomal packing for two-dimensional superdisks for both concave and convex shapes was presented by Jiao et. al [2009]. An algorithm to generate ordered and disordered sphere packings using linear programming was proposed by Torquato and Jiao [2010]. This method is termed as Adaptive shrinking cell formulation and is based on sequential linear programming (SLP) techniques. The packing space is divided into cells and this method involves displacement of particles, and deformation and volume changes to the cell. The result is a jammed packing. Weller and Zachmann [2010] proposed a packing algorithm for arbitrary objects that was based on the concept of "Apollonian sphere packing". Apollonian packings are known to be space filling [Weller and Zachmann, 2010]. The approach is based on fitting the largest sphere inside the object and recursively inserting more non-overlapping spheres that are confined within the boundary of the object. The drawback of this algorithm is that it yields an approximation of the object's medial axis because of the way it is constructed. Lagarias and Zong [2012] present an interesting case study that discusses the mysteries in packing regular tetrahdron. Various algorithm dating from old to new are described, compared and their problems are discussed.

3 Technical background

The various algorithms proposed for packing of particles can generate ordered and disordered arrangements. Most of the ordered arrangements, specially lattice packing is known to generate denser outputs, but it has been proven that some disordered packings can generate equally denser packings or even higher. Most of the packing algorithms, proposed in different fields, are mainly based on packing of spherical particles. Some algorithms have been proposed for packing other geometries such as cylinders, platonic and Archimedean solids. Most of the approaches which are based on non-spherical shapes are based on lattice vectors and yield to ordered packing. Packing of arbitrary shapes is still not much explored.

Lattice based packings are easier to implement and provide faster results. On the other hand, random packings are time consuming and also require lot of computation. Both types of packing have their usage and existence. Random behaviour is more natural in existence, and thus the use of random packing as the basis can lead to visually realistic outcomes. The algorithms mainly use sphere because of its uniform shape, which makes it easy to compute and detect overlaps or collisions. Using non-spherical shapes for particles makes the algorithm time-consuming and complex. For non-spherical shapes, Lattice based packing is easier to implement because only the positions and rotations need to be calculated unlike Random packing in which overlap needs to be detected. The position of the particles can be easily calculated based on lattice vectors, especially if the shapes are centrally symmetric, it makes the arrangement and calculation less tedious.

For non-lattice packings, specifically in case of random packing, it is easier and faster to detect overlap for spheres than for arbitrary shapes. The detection of collision can be really exhaustive depending on what shape is selected for the particle. If any arbitrary shape has to be used for the particle, an alternate way of detecting overlap could be to check the bounding box values of the particle and the neighbouring particles. Even if the arbitrary shapes are of different sizes, the bounding box values can be calculated and used for overlap detection. This is an efficient way of detecting overlaps.

3.1 Lattice packing

The packing is termed as lattice packing when there are spheres with centers u and v, that means there are also spheres having centers u+v and u-v, considering 0 to be the center of the packing [Conway & Sloane 1993]. The lattice packings are all based on the same concept, where the lattice vectors decide the positions of the particles in the arrangement. This arrangement is formed in layers, where a particle layer is created in a particular formation and then another layer of particles is laid on top of this layer. This process is repetitive and results in ordered packings.

Not all the layers are arranged in the same way, but the arrangement does repeat itself. So depending on which packing type is chosen, every alternate or third layer may be same. Numerous arrangements can be formed by making use of this method. The formation or layout of the particles in the layer is dependent on the type of lattice chosen, and there are multiple variations possible.

Bravais lattice packing is one of the sub-categories of lattice packing, in which the centers of the spheres form an additive group. The term Bravais lattice packing is commonly used in the field of crystallography. Bravais lattices can fill up the whole space, when the packing pattern is repeated. Each lattice point, defined by vector r can be obtained from the following equation [Zeghbroeck 1997]:

$$r = ka_1 + la_2 + ma_3$$

where, k, l and m are integers and a_1, a_2 and $a_3$are unit vectors.

3.1.1 Octahedral structure

O_0- lattice packing

Face centered cubic packing is one of the forms of Bravais lattice. Assuming that the packing space is divided into cubic cells, then this structure is formed by placing spheres at the cube corners as well as in the center of each face of the cube. If all of the corner spheres are removed, the remaining six spheres form a hexagonal array. These remaining six spheres are tightly packed, leaving no scope of packing them closer than that. If all these spheres are connected to each other, they result into eight planes that forms an octahedron, thus they are called 'Octahedral' planes [FCC, BCC and HCP Metals n.d.]. The structure of this packing is ABCABC.

This optimal packing is defined as O_0 lattice packing by Jiao et al. [2009]. The layers are stacked in a way that the spheres in the top layer fit in the holes created in the bottom layer. Each sphere contacts twelve neighbours in total, consisting of four neighbours in each layer: own, bottom and top. The lattice vectors are given by the following equations [Jiao et al. 2009].

$$e_1 = 2i$$

$$e_2 = 2j$$

$$e_3 = i + j + 2(1 - 2^{1-2p})^{1/2p}k$$

The spheres can be replaced with octahedra, in that case the planes of the two layers coincide with each other. The packing density, $\phi = 0.7959$ approximately, which is dependent on the size of the particle.

O_1- lattice packing

The densest known packing of regular octahedra is achieved by Bravais lattice packing [Jiao et al. 2009], and was discovered by Minkowski [1904]. In this packing, each octahedra contact 14 others. The packing density, $\phi = 18 \ / \ 19 = 0.9473$, which is definitely denser than the one described in the section above.

The Bravais lattice vectors for this packing are given below [Minkowski 1904] [Betke et al. 2000].

$$a_1 = \frac{2}{3}i + \frac{2}{3}j - \frac{2}{3}k$$

$$a_2 = -\frac{1}{3}i + \frac{4}{3}j - \frac{1}{3}k$$

$$a_3 = \frac{1}{3}i - \frac{1}{3}j - \frac{4}{3}k$$

3.1.2 Hexagonal close packing

HCP is a periodic packing and is considered as one of the important packing structures, in ranking with face-centered cubic and body-centered cubic packings. Though not a Bravais lattice in itself, this packing method is based on simple hexagonal Bravais lattice as its underlying structure, that is generated by stacking two-dimensional triangular nets placed on top of each other. The lattice vectors are given by the following equations [FCC, BCC and HCP Metals n.d.].

$$a_1 = ai$$

$$a_2 = \frac{a}{2}i + \frac{\sqrt{3}a}{2}j$$

$$a_3 = ck$$

where, c is the distance between two layers and a is an integer.

The first layer is a close-packed triangular lattice, followed by the second layer in which the particles are placed in the holes created by the triangles in the first layer. The third layer is formed by placing the particles in the depressions created by the second layer and so on. This means that the arrangement of third layer is same as the first layer. So in general, alternate layers are same. The resulting packing structure is ABABAB.

The distance between two interpenetrating hexagonal Bravais lattices which hcp is based on, is given by the following value [FCC, BCC and HCP Metals n.d.]:

$$\frac{a_1}{3} + \frac{a_2}{3} + \frac{a_3}{2}$$

3.1.3 Optimal lattice vectors

To generate dense packings of nonoverlapping, nontiling polyhedra within an adaptive fundamental cell that is subjected to periodic boundary was called the adaptive shrinking cell ASC scheme [Torquato & Jiao 2009]. The densest packing of the platonic shapes, except tetrahedron, are the Bravais lattice packing. The values of packing density obtained by ASC scheme are at par with the density obtained from Bravais lattice. An unsaturated packing of particles is generated within the cells as an initial condition, the positions and rotations of the polyhedra are design variables which need to set appropriately for optimization. The cell can also be deformed or compressed/expanded, which is the reason for calling these cells as adaptive fundamental cell.

Packing of cubes

Each cube in the packing contacts 26 others, which includes vertex-to-vertex contacts, and the packing density is given by $\phi = 1$. The optimal lattice vectors are given as [Torquato & Jiao 2009]:

$$a_1 = (2, 0, 0)$$

$$a_2 = (0, 2, 0)$$

$$a_3 = (0, 0, 2)$$

Packing of dodecahedra

As an initial condition, unsaturated random packings with densities 0.15 to 0.3, unsaturated simple cubic and the optimal lattice packings with densities from 0.3 to 0.6 have been tested. The maximum packing density is achieved when an unsaturated optimal lattice packing with density 0.72 as an initial condition is applied [Torquato & Jiao 2009].

Each dodecahedron of the packing contacts 12 others. The packing density is $\phi = (2 + \Phi)/4 = 0.904508....$The optimal lattice vectors are given as [Torquato & Jiao 2009]:

$$a_1 = [2/(1 + \Phi), 2/(1 + \Phi), 0]$$

$$a_2 = [2/(1 + \Phi), 0, 2/(1 + \Phi)]$$

$$a_3 = [0, 2/(1 + \Phi), 2/(1 + \Phi)]$$

where, $\Phi = (1 + \sqrt{5})/2$ is the golden ratio.

Packing of icosahedra

Various initial conditions were tried to achieve the dense packings of icosahedra. The different techniques used include unsaturated random packing with density 0.2 to 0.3, unsaturated lattice packings such as bcc and fcc, and the densest lattice packing with density 0.3 to 0.65 [Torquato & Jiao 2009].

Each icosahedron of the packing contacts 12 others and the packing density is $\phi = 0.83637....$ The optimal lattice vectors are given as [Torquato & Jiao, 2009]:

$$a_1 = \frac{2}{(1+\Phi)} \begin{pmatrix} (-33/8 - 39\sqrt{5}/8)x^2 + (39/4 + 33\sqrt{5}/4)x - 11/4 - 3\sqrt{5}/2 \\ (-1/4 - \sqrt{5}/4)x + 1 + \sqrt{5}/2 \\ (33/8 + 39\sqrt{5}/8)x^2 + (-19/2 - 8\sqrt{5})x + 13/4 + 3\sqrt{5}/2 \end{pmatrix}$$

$$a_2 = \frac{2}{(1+\Phi)} \begin{pmatrix} (-39/8 - 33\sqrt{5}/40)x^2 + (35/4 + 41\sqrt{5}/20)x - 5/2 - 23\sqrt{5}/20 \\ (5/4 + \sqrt{5}/4)x - 1 - \sqrt{5}/2 \\ (-39/8 - 33\sqrt{5}/40)x^2 + (15/2 + 9\sqrt{5}/5)x - 3\sqrt{5}/20 \end{pmatrix}$$

$$a_3 = \frac{2}{(1+\Phi)} \begin{pmatrix} (3/2 + \sqrt{5}/2)x - 2 - \sqrt{5} \\ x \\ 0 \end{pmatrix}$$

where, $\Phi = (1 + \sqrt{5})/2$ is the golden ratio, and $\bar{x} \epsilon \{1, 2\}$ is the unique root of the polynomial.

It is found that $\bar{x} = 1.591\ 603\ 01...$ and therefore the lattice vectors, up to nine significant figures, are given as below.

$$a_1 = (0.711782425, 0.830400102, 1.07585146)$$

$$a_2 = (-0.871627249, 0.761202911, 0.985203828)$$

$$a_3 = (-0.06919791, 1.59160301, 0)$$

3.2 Non-lattice packing

Most of the packings are not lattice packings but there still can be packings that can be as dense as lattice packings like fcc and hcp or even denser. The lattice packing is built up in layers. Considering the example of fcc or hcp, the first layer of spheres is created by placing the sphere centers in points marked a, then the next layer can be created by positioning the spheres in 'b' or 'c' layout. If the spheres are placed at 'b' positions, the next layer can be created by placing the spheres in 'a' or 'c'. The resultant will be fcc packing if the layers are arranged in either of the following manner: 'abcabcabc' or 'acbacbacb' [Sloan & Conway 1993]. The 'abababab' arrangement of the layers will result into hcp packing.

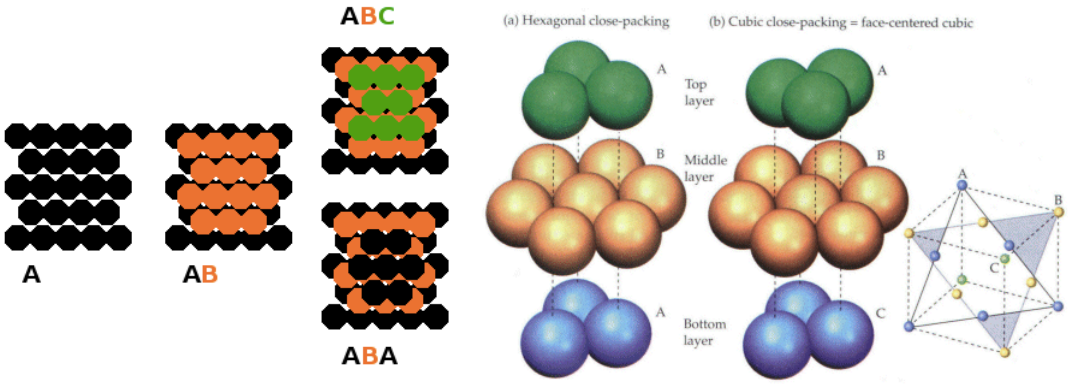

Figure 1: Different layers of packing (left) [. Layers of packing for hcp and ccp(right) [Simple modes of packing n.d.]

But, there are numerous non-lattice possibilities which will result into the same density, such as 'acbabacb' or 'abcacabc'.

Random packing is an interesting type of non-lattice packing and there can be many ways to achieve it. Two different methods of random packing are discussed in this section. The mesh is divided into cells which makes it faster to detect collision or overlap, which makes it easier to handle overlap detection. If the mesh is considered as one complete geomtery without any cell divisions to check for overlaps, it will too time consuming as every particle will need to be validated against all the particles existing in the mesh. The size of the cell can be varied, the size of the cell needs to have an optimum value, it should neither be too small, nor too big. When the particle is created it is assigned to the cell to which it belongs which is decided by the position of the particle.

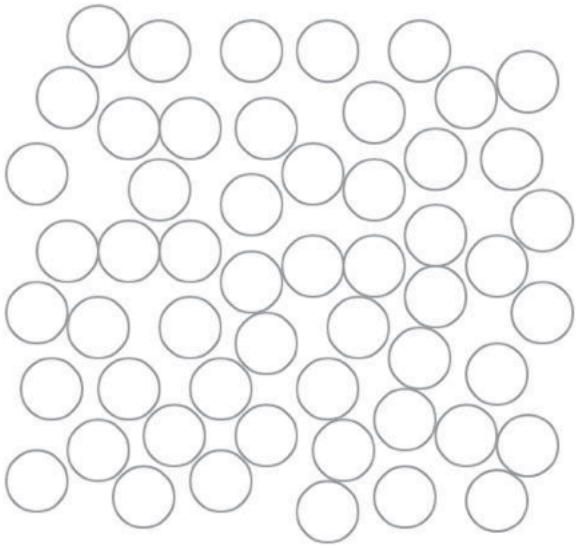

Figure 2: Aperiodic packing [Hopkins et al. 2012]

3.2.1 Binary packing

The packing space can be formed by generating N_s points of type S and N_L points of type L, which will combine both small(S) and large(L) points [Hopkins et al. 2012]. Each pair of type S must be separated from each by a minimum distance of $2R_s$ and each pair of points of type L must be separated by a minimum distance of $2R_L$. And if the pair includes one point of type S and the other point of type L, then the minimum distance between these two points should be $R_s + R_L$.

A variation of the lattice packing can be a periodic packing. A periodic packing in this case can be obtained by placing a fixed number of particles in one cell, which can be called the unit cell. Since the cells are all of the same size, the particles can be replicated to all other cells. The particles can be of different sizes. The packing fraction for such a packing of spheres is given by [Hopkins et al. 2012]:

$$\phi = \frac{\Sigma v(R_i)}{V_U}$$

where R_i is the radius of each sphere in the cell, and V_U is the volume of the cell. No binary packing can be a lattice packing since the basis of binary packing consist of atleast two particles.

Aperiodic packing is the one in which there is no long-range order, which means that the minimal basis should be equal to the number of particles. For a

packing with large number of particles, it can be aperiodic, if there are very few repetitions of the minimal basis [Hopkins et al. 2012].

For achieving jamming conditions in such packings there can be various techniques possible. One such method is called "host-guest" packings. In this technique, a subset of the packing (usually larger spheres) are packed as a jammed periodic packing. The smaller spheres then occupy the space formed by the interstices by the larger spheres. In this case the smaller spheres do not contact the larger spheres, and can be moved in the packing domain, without affecting the overall density.

An additional feature for allowing the particle to grow has been implemented. If the growth is enabled, the particle will grow in size so that it touches its nearest neighbour. The growth parameter δ_i can be given as [Kansal et al. 2002]:

$$\delta_i = \delta\sigma_{i,0}$$

where $\sigma_{i,0}$ is the initial diameter of the sphere.

3.2.2 Linear packing

The mesh is divided into cells and the particles are created inside those cells. Starting from an initial unsaturated configuration of particles of fixed size in the cell, the positions and orientations are the design variables for optimization [Torquato & Jiao 2010]. The initial state of linear programming is a setup with some initial particle configuration, and involves a deterministic collective motion of the particles to achieve a higher density.

When a displacement to the particle position is applied, the new particle position is given by the following equation.

$$x = x + \Delta x$$

Random packing of unequal spherical particles can be generated using Monte Carlo method [He et al. 1999]. The particle radii obeying a given distribution are generated and randomly placed within the packing domain. The packing density is kept high when generating the particles. As the next step, a relaxation step is performed which will relocate the particles to new positions based on its neighbour positions and radii. This relaxation step is performed to minimize or elimate overlaps.

The process can be summarized in high-level steps below:

1. Initial particle size generation
2. Initial position allocation
3. Overlap relaxation

4. Packing space expansion

Since the packing space is bounded by the mesh, the packing space expansion is not possible, because even the shape of the packing space is arbitrary. As a modification to the step no. 4 above, a radius relaxation step is performed, which will be described in detail in the Section 5.

The overlap rate of two particles with radii r_i and r_j, with d_{ij} as the distance between the centers of these two particles, is defined as [He et al. 1999]:

$$(r_i + r_j - d_{ij})/(r_i + r_j)$$

For each particle i, a search is conducted for particles that overlap particle i, then from each of the overlapping particles, j, a new position can be calculated by the following equation [He et al. 1999]:

$$R_{ji} = R_j + (R_i - R_j)\frac{(r_i + r_j)}{d_{ij}}$$

where R_i and R_j are vectors of the centers of particles i and j, respectively. If the particle i is overlapped by n particles, then the new position of the particle is given by [He et al. 1999]:

$$R_i = \frac{1}{n_i}\sum R_{ij}$$

4 Design

The solution is designed in C++ language. The libraries used include NGL and some of the QT functions are used. "SignedDistanceFieldFromMesh.h" [Sanchez 2011] header file is used to check whether the point is inside the mesh or not. The aim was to implement various types of packings. Also, not only the most commonly used shape for particle packing i.e sphere has been used, but packing has been implemented for other shapes as well. Lattice packing provides an option for platonic shapes, whereas Random packing gives an option to select OBJ mesh as the particle.

Lattice packing has been implemented both for spherical particles as well as some platonic shapes. The advantage of using lattice packing is that it is efficient as overlap detection is not needed. The position of the particles are calculated by using the lattice vectors and is based on the size of the particle. The particles are stacked in layers, one layer is created and then another one is added on top of it and this recursive process continues as long as the center of the particle is within the boundary of the bounding box.

Random packing is implemented for spherical particles and also any OBJ file can behave as the particle. Since, it is complex to deal with collision for particle shapes other than spheres, more features are added on sphere packing. Particle positions are generated randomly using size of the container's bounding box as the basis. Binary packing is one of the choices, in which the packing is created in two iterations. A first cycle of particle creation executes for the provided parameter values, followed by the second one. A third iteration to generate particles can be added by enabling the "ternary" parameter. The reason for having multiple levels is that the values mainly the number of particles and radius can vary. The result of this packing will be a mix of particles with different radii, with the approximate number of particles defined. The particles added in any of the iterations have a base radius and this radius will only change (grow) if the growth option is selected for that iteration. The size or value of growth is determined by its distance to the closest neighbour. For any particle, the distance to its nearest neighbour is calculated, and the particle is scaled by that value. So, after the particle grows, it will touch its nearest neighbour. As the particle size is allowed to increase beyond its initially specified radius, this will enable the particle to occupy more space in its surrounding resulting in a denser packing.

When the particles are created, the point position of the particle, or the center of the shape, should lie within the boundary of the mesh or packing space. The particles near the boundary may partially exceed the boundary. The extent to which the particle lies outside the boundary, depends on its closeness to the boundary and the size, which would be the radius in case of spheres. The particle position is checked, and if the particle center lies within the mesh the

particle is created. In some cases, an additional option is provided to generate the particles which lie completely within the boundary of the mesh. Because of the uniform shape of the sphere, this option is applicable to spheres only.

The other type of Random packing implemented is called Linear Packing. In this packing technique, the particles (spheres) at random positions are created, even if they overlap. After the particles are created, a relaxation is performed on all the particles. This relaxation process calculates the new position of the particle based on the positions of its neighbours. Since the spheres were still found to be overlapping with each other significantly, an extension is added which re-calculates the radius of the particle, based on its neighbours.

4.1 Algorithm

The high-level algorithm is described below, but more details are provided in the Section 5.

1. Select a default shape from the pre-saved OBJ files or import OBJ using file browser.
2. Select the Mode of packing: Lattice or Random
3. Select the Type of packing, the options depend on the Mode and the shape of the particle chosen. For Lattice, options are: Cubic, Hexagonal or Octahedral.
4. Select the shape of the particles.
5. Calculate the bounding box of the OBJ mesh.
6. The point position, whichever be the Mode of packing, is generated within the limit of the bounding box, and is tested for belonging inside the mesh or not.
7. For Lattice Packing:
   (a) Depending on the Type chosen, generate the packing based on the lattice vectors.
   (b) For packing of spheres, in hcp or ccp arrangement the horizontal layer is generated first then another layer is stacked on top of the one created previously.
   (c) For packing of other platonic shapes, the lattice vectors will not create a horizontal layer, so in this case, one layer is generated using X and Z components first. Then the next layer is added using the height specified Y component.
   (d) The particle positions are calculated based on the lattice vectors, and is verified to fall within the boundary of the mesh. If the position lies inside the particle is created, and is discarded otherwise.

8. For Random Packing

   (a) The options for particle shapes are: Sphere, Cube or OBJ. Some pre-saved OBJ particle shapes are provided, but any OBJ can be chosen for the shapes of the particle.

   (b) Divide the mesh into cells.

   (c) Generate a random position of the particle.

   (d) Based on the position and radius check if the particle overlaps with any existing particle. The logic for detecting overlap is covered in detail in the Section 5.2.

      i. If the particle does not overlap any of its neighbours, create the particle.

      ii. If the particle overlaps with any other particle, then try a different position of the particle.

   (e) For Binary/Ternary packing,

      i. The above step is performed in iterations. For Binary, it will be two iterations and for Ternary, the number of iterations will be three.

      ii. The maximum number of particles and the number of retries can be setup for each iteration.

      iii. If the 'Grow' option is selected for any iteration, then after all the particles in that iteration are created, the size of the particles created in that iteration is increased, until it gets in contact with its closest neighbour.

4.2 Class Diagram

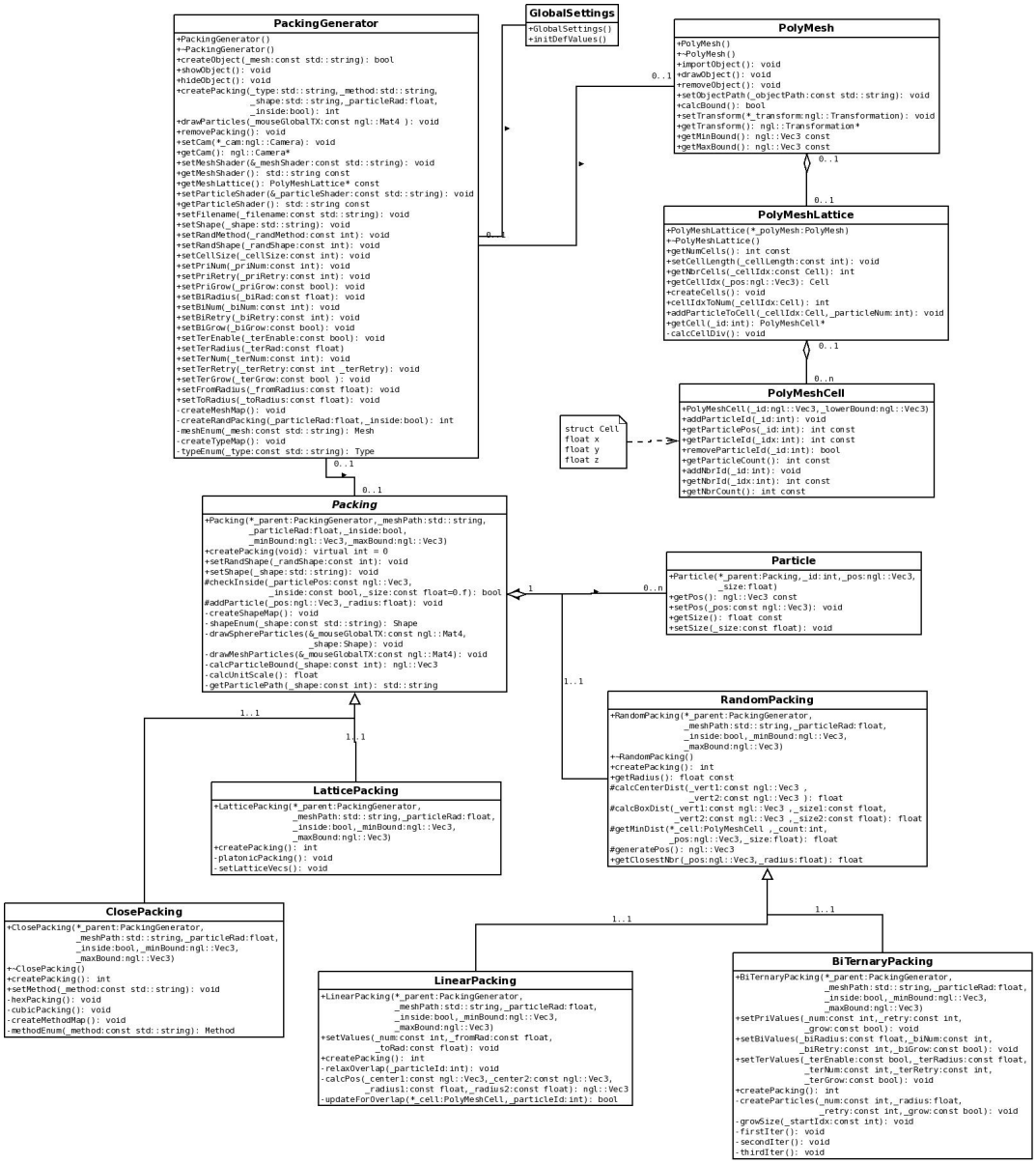

Figure 3: Complete class diagram

The class **MainWindow** handles the user interface and connects the signals for the controls. This class sets the default values of some controls. It will enable or

disable some controls based on the values in other controls. The **GLWindow** class acts as the mediator between **MainWindow** and **PackingGenerator** class. It also loads the shader, setup the the camera and light.

PackingGenerator is the core class that connects and co-ordinates with the other classes. This class is the one that is called from **GLWindow**. This class is responsible for transmitting information to the respective classes for generating the desired output. Based on the options chosen for packing mode and type, the respective methods are invoked.

PolyMesh class creates the mesh, which is the container for bounding the particles. This mesh is treated as the boundary for particle packing space. For lattice packing, since the packing is sequential the mesh need not be divided into cells. In case of Random packing, if the mesh is not divided into cells, then each time a random position for the particle is generated, it has to be checked for overlap against all existing particles. This may take a substantial amount of time depending on the number of particles existing already. As the number of particles increase, this process will become slower. An efficient way to check the overlap is to divide the mesh into cells so that overlap has to be detected in the current cell and neighbouring cells.

Packing is an abstract class and is the base class for the classes that generate different types of packing. This class will add particles to the **Particle** class. It will also draw the particles on the screen depending on the type the particle. Packing is the base class for three classes: **ClosePacking**, **LatticePacking** and **RandomPacking**. **ClosePacking** class generated hcp and ccp packing structures for spheres. The packing types in this class are also defined by lattice vectors. But this class has been separated for maintainability. **LatticePacking** is used to generate packing based on lattice vectors. This class will generate packing for platonic shapes and the packing arrangement generated through this class is layered. The particle positions are calculated based on the lattice vectors depending on the type of packing.

RandomPacking class functions as the base class for two classes. The first one is **BiTernaryPacking**, which will generate packing in two or three iterations. The main purpose of having packing in iterations is to make it more scalable and flexible. Each iteration will create particles with the specified size. Additional options are provided to control the maximum number of particles and number of retries. This type of packing will result in particles with different sizes. An additional feature to grow the particle is included, which will scale the size of the particle till it gets in contact with its closest neighbour. This option will not only result in particles with varying sizes, but will also increase the density of the packing. The second class is **LinearPacking,** which generates the particles at random positions. Then, a relaxation step is executed on the overlapping particles. Based on the position and radius of each of its neighbouring particle,

a new position is then evaluated for the overlapped particle. Since the particles still overlap considerably, it becomes important to modify the radius of the particle. Similar to the calculation of new position, the new radius is calculated based on the position and radius values of its neighbours.

The **PolyMeshLattice** and **PolyMeshCell** classes are used in case of random packing. **PolyMeshLattice** class is used to divide the mesh into cells. The bounding box of the mesh is evaluated and the cells are divided based on the bounding box values. The size of the cell is provided as a parameter in the interface. **PolyMeshCell** class holds the data specific to each cell, like the cell index and the lower bound of the cell which is the position of the lower division of the cell in world co-ordinates. **PolyMeshLattice** class calculates the neighbouring cells for all the cells that have been created. The reason to store all the neighbouring cells in advance, is because the cell size is fixed initially when the mesh is divided, so the neighbours will not change. Also, as multiple particles can be created in one cell, it will be an extra effort to find the neighbouring cells again and again, for all point positions which belong to the same cell (the point position can be used as a basis to find the cell it is located in or specifically the cell index).

Particle class will hold the information specific to the particle like particle ID, position and size.

Maps have been used in multiple scenarios. To map the associative path for the mesh/shape of the particle, the names which are strings are mapped to the OBJ path. Another instance is, to be able to use string in switch statements. Maps are created for storing the associations and the equivalent values are fetched using respective methods.

5 Implementation

Different methods have been implemented to achieve various forms of arrangement of particles. The methods used for particle packing can be roughly divided into two categories: lattice and random packing. To achieve lattice packing for spherical particles, the commonly known close-packing methods such as hexagonal close-packed (hcp) and face-centered cubic (fcc), also known as cubic close-packing (ccp) are applied. Packing of some other platonic shapes is achieved by using lattice vectors. Packing has also been designed for similar and different sized particles.

5.1 Lattice packing

Lattice packing has been generated for spherical as well as non-spherical particles. For spherical particles, hcp and ccp are the two packing types implemented. For non-spherical particles, the packing is based on lattice vectors. The point positions are generated using the lattice vectors. This type of packing is generated in layers and is repetitive, some layers are replica of the others. In most of the case every second or third layer will be same. All the lattice vectors used are provided in Section 3.1.

The high-level steps for lattice packing are given below:

1. Generate point position based on the lattice vectors
2. The particle position is only generated within the bounding box of the packing domain.
3. The point position is then checked if it lies within the mesh
   (a) If the point position is inside the mesh, create the particle.
   (b) Discard the position if the point lies outside the mesh and generate a new position.

5.2 Random packing

Random packing has been implemented mainly for spherical particles as well as arbitrary shapes. For packing of spheres, two different approaches are used. The underlying structure of these methods is the same and most of the functionality used is common and applies to both. Both the approaches are described below and rest of the section will cover the common aspects. Both these approaches are capable of packing unequal spheres.

5.2.1 Binary/Ternary Packing

This approach is based on the generation of multiple iterations of packing. Two (Binary) or three (Ternary) iterations can be selected for generating the packing.

sectionEach iteration contains the number of particles, radius, number of retries and option for growth. The radius in the iteration is common for all the particles generated in that iteration. In this approach, the position is randomly generated within the bounding box of the mesh, and then is checked whether it lies inside the mesh or not, the position is only considered valid if it lies within the mesh. This newly generated position is validated for overlap with any of neighbouring particles, if it fails then a new position is re-tried.

There is a limit to the number of times the process is repeated, either until the maximum particle count is reached or the maximum number of re-tries for the same n^{th} particle is reached. The idea behind placing a validation for maximum re-tries for the same particle, is that if x number of random positions have been tried and the particle still fails to find a position, then the possibility is that not enough space is available for the particle to find a space. Had there not been a limit on maximum number of re-tries then the algorithm would continue to search for a position until the maximum number of particles are attained, so in case of not enough space, this may result into an endless loop. The particle is only inserted when it finds a valid position.

5.2.2 Linear Packing

This approach is somewhat similar to the first approach. The particle positions are randomly generated within the mesh. The difference is that the randomly generated position is not validated with the neighbours immediately, and the particle is inserted. This will lead to overlaps of particles. After the particles are generated, for all the particles which overlap with its neighbours, a relaxation is provided and a new position for the center is found. This approach does not have iterations as the first one and the radius of the particle is not fixed. Each particle will have a different radius, which is a random value. So not only the position of the particle is random but also the radius of the particle is random, but to limit the radius within a limit, a range is defined.

5.2.3 Collision detection

The particles inside the mesh need to be non-overlapping. In case of random packing, it is of prime importance on how the collision detection is done. As mentioned earlier, collision detection for non-spherical shapes can prove to be exhaustive. Also, if the shape is arbitrary, simple techniques to detect overlap may no longer work. One of the ways to detect overlap, can be to check the vertices/edges of one particle, and if those fall within the boundary of another particle, the particle cannot be created at that position. For packing of arbitrary shapes, digitization approach was mentioned by Jia and Williams [2001], that was based on digitization of particle shape as well as the mesh. This method requires a lot of computation for finding the cells which are filled by the particle.

An alteration to the approach, can be dividing the mesh into cells with size of bounding box. The approach basically can hold one particle in one cell, so when a new particle is created, the only check to be made is whether the cell is occupied already. This approach works well for structured packing, but it cannot be implemented for random packing as such, since the particles in the same layer will always be at the same level and this is more like lattice packing. If the cell size is bigger than the bounding box of the particle, then the particles can be positioned at different levels in the same layer, but the rest of the space in the cells will always be unoccupied.

To make the packing of arbitrary shapes more compact and random, the cell size is considered irrespective of the bounding box size. Any number of arbitrary shaped particles can be added to a cell, as long as they do not overlap each other. The lattice cells are still the normal cell divisions mentioned previously as part of the algorithm in the Section 4. Intersection of two particles is controlled by finding the distance to all the particles from the neighbouring cells. The approach used is same as for spherical particles, the only difference lying in the calculation of the nearest neigbour. While in case of spheres, the radius is used for detecting overlap, whereas for arbitrary shapes bounding box values are used for overlap detection.

The following steps are used for overlap detection:

1. Check all neighbouring particles, which includes particles in its own cell as well as the neighbouring cells.
2. To find the distance of the particle to its closest neighbour, the calculation is based on the following particle and neighbour values:
   (a) Spheres - Calculation is based on position and radius. The center to center distance is calculated and checked against sum of radius.
   (b) Arbitrary shapes (OBJ) - position and bounding box. Overlap is checked in X, Y and Z axis and the minimum value is considered.

   [Note: These values are stored at particle level.]
3. In case of overlap,
   (a) a new random position is tried.
   (b) the number of retries keeps increasing unless a non-overlapping particle position is found.
4. For no overlap, the particle is created.
5. The particles of the specified radius are inserted.
6. If the growth option is selected, then the particle is scaled uniformly by the minimum distance to the closest neighbour.

6 Results

This section shows the results generated for different types of packing.

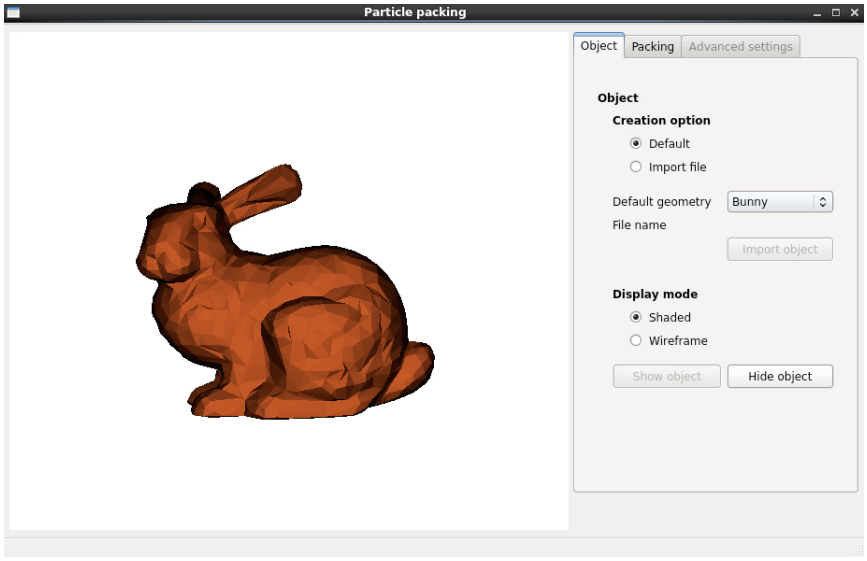

Figure 4: User settings

6.1 Close packing

6.1.1 Hexagonal close packing

Hexagonal closed packing has been generated for two different meshes.

Figure 5: Hexagonal close packing

6.1.2 Cubic close packing

Cubic closed packing has been generated for two different meshes.

Figure 6: Cubic close packing

6.2 Lattice packing

Lattice packing is generated for different platonic shapes: cube, dodecahedra and icosahedra.

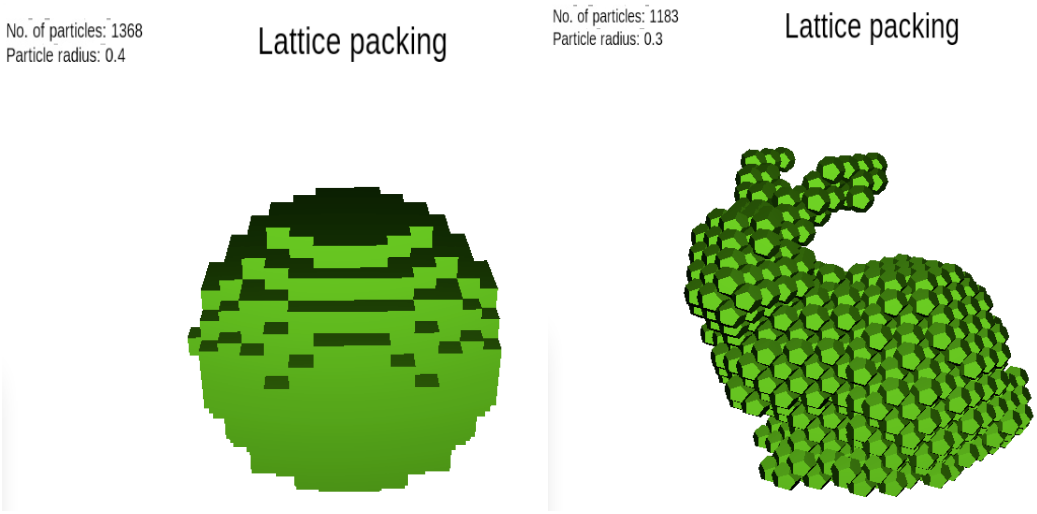

Figure 7: (a)Lattice packing generated for cube particles. (b) Lattice packing for dodecahedron.

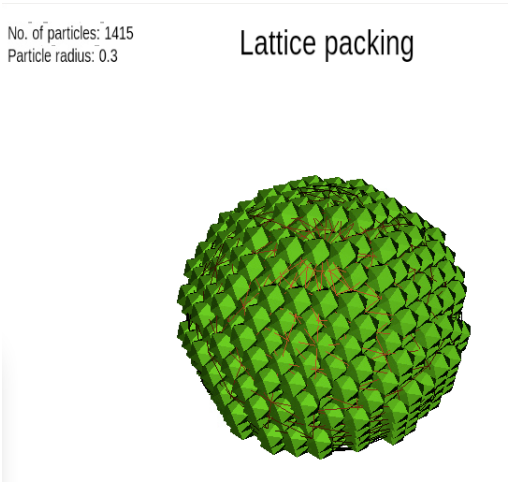

Figure 8: Lattice packing for icosahedron.

6.3 Random packing

6.3.1 Binary/Ternary packing

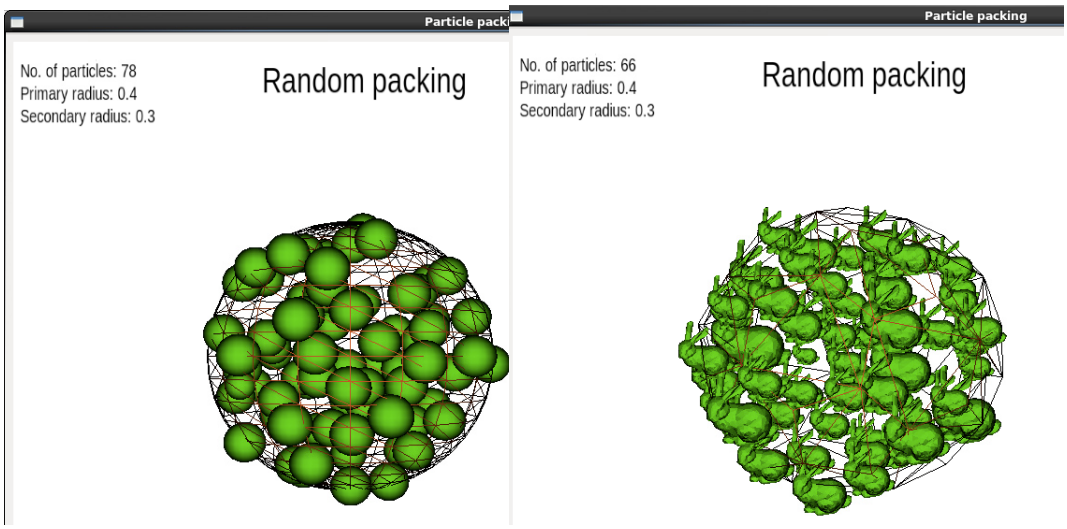

Figure 9: Binary packing using different shapes of particles (OBJ)

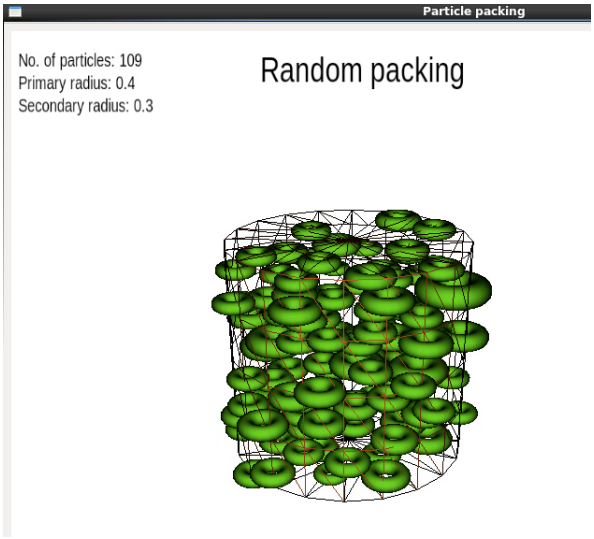

Figure 10: Binary packing with growth parameter selected.

6.3.2 Linear packing

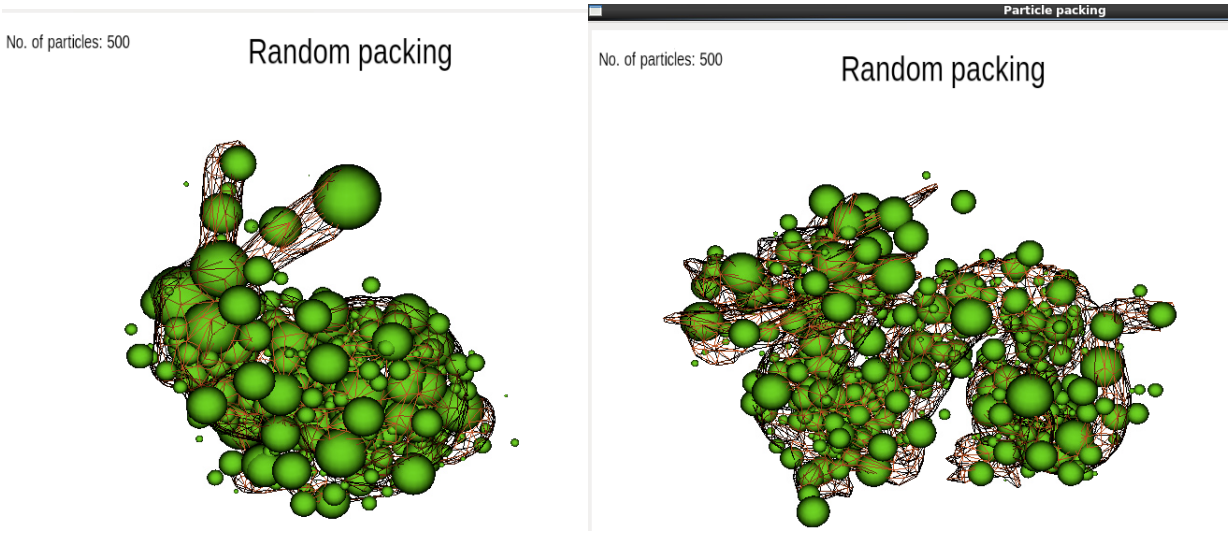

Figure 11: (a)Lattice packing generated for cube particles. (b) Lattice packing for dodecahedron.

7 Conclusion and future work

Packing of particles in multiple arrangements has been implemented. Lattice packing has been implemented for spherical and platonic shapes. Random packing has been achieved for spheres and cubes. Random packing has been extended to use OBJ files as shapes for particles, in which some pre-saved OBJ files are provided as choices. An additional option of importing any OBJ mesh for the particle shape has been provided. This packing can be generated for all particles of the same size, but additional options are provided to have different sizes of the particles. A parameter for growth is also added, that increases the size of the particle so that more space is occupied and this will lead to more denser outcomes.

Multiple algorithms are implemented, which are based on different technical papers. Some of the features from different papers are combined and implemented as one algorithm. Also, some new ideas have been implemented like provision of growth option for Binary/Ternary packing. As an attempt to work on packing of arbitrary shapes, another idea of using bounding box as the shape for particles have been implemented, in which the particles are OBJ files.

Lattice packings are faster to compute as overlap doesn't need to be calculated. Thus, it can produce faster results. The non-lattice packings which are layered, also produce structured arrangements. So in general, packing in layers is easily achievable and also produces denser results. Random packing, on the other hand, which also falls under the category of Non-lattice packing, requires more computation and will take longer to generate results. The different types of packing implemented as part of the project, produce results in fair amount of time. Binary/Ternary packing produces results with different sphere sizes, but since the particle positions are fixed, the packing space cannot be fully utilized. The results from Linear packing technique provide denser packing as compared to the Binary method of packing.

Particle packing finds applications in many areas, and can act as the basis of different simulations. One of the uses could be for water or sand simulations. Another one is to make use of the particle packing information for point based dynamics.

There are hardly any algorithms that describe packing of arbitrary shapes densely, but given more time this aspect could be delved into deeper. Instead of having all particles of the same shape, particles with different shapes can be an added feature too. An additional option of exporting the packing generated in C++ as OBJ can prove to be useful, as that can be used by other program as an input geometry for simulations.

References

Betke, U., & Henk, M., Comput. Geom., 16, pp. 157-186, 2000. Conway, J. H., Sphere Packings, Lattices and Groups , Springer, 1993.

Donev, A., Torquato, S., Stillinger, F. H., "Neighbor list collision-driven molecular dynamics simulation for nonspherical particles: I. Algorithmic details II. Applications to ellipses and ellipsoids", J. Comp. Phys, 202 (2), pp. 737–764, 765–793, 2005.

"FCC, BCC and HCP Metals", Accessed August 2015, <http://che.uri.edu/course/che333/Structure.pdf>, Pg 1.

He, D., Ekkere, N. N. & Cai, L., "Computer simulation of random packing of unequal particles", Physical Review, 60 (6), 1999.

Hopkins, A. B., Stillinger, F. H. & Torquato, S., "Densest binary sphere packings", Physical Review, 85, pp. 1-19, 2012.

Kansal, A. R., Torquato, S. & Stillinger, F. H., "Computer generation of dense polydisperse sphere packings", Journal of Chemical Physics, 117 (18), pp. 12-18, 2002.

Jia, X., & Williams, R. A., "A packing algorithm for particles of arbitrary shapes", Powder Technology, 120, pp. 175-186, 2001.

Jiao, Y., Stillinger, F. H. and Torquato, S., "Optimal Packings of Superdisks and the Role of Symmetry Physical Review Letters", 100, 245504, 2008.

Jiao, Y., Stillinger, F. H. & Torquato, S., "Optimal packings of superballs", Physical Review, 79, pp. 1-12, 2009.

Lagarias, J. C. & Zong, C., "Mysteries in Packing Regular Tetrahedra", Notices of the AMS, 59 (11), pp. 40-49, 2012. Conway, J. S. & Torquato, S., "Packing, tiling, and covering with tetrahedra", PNAS, 103 (28), pp. 12-17, 2006.

Minkowski, H., Nachr. Ges. Wiss. Göettingen, Math.-Phys. Kl- .Dichteste gitterförmige Lagerung kongreuenter Körper, pp. 311, 1904.

Sanchez, M., "Continous signed distance field representation of polygonal meshes", Accessed August 2015, <http://nccastaff.bournemouth.ac.uk/jmacey/MastersProjects/MSc11/Mathieu/msanchez-sdf-thesis.pdf>

Sarto and Van Zeghbroeck, B., "Photocurrents in a Metal-Semiconductor-Metal Photodetector", IEEE Journal of Quantum Electronics, 22, pp. 88-94, 1997.

“Simple modes of packing”, Accessed August 2015, <https://cluster13-files.instructure.com/courses/986898/files/31768395/course%20files/006%20Liquids%2C%20Solids%2C%20%26%20Intermolecular%20Forces/Close%20Packing/Close_Packing.htm?download=1&inline=1&sf_verifier=&ts=&user_id=&verifier=ojbHMiy731x7omWoKKQPpgtkyGWdezTzMsqFWWDX>

“Some important examples of crystal structures with lattices of bases”, Accessed August 2015, <http://materias.fi.uba.ar/6210/Ap%C3%A9dice%203.pdf>

Torquato, S. & Jiao, Y., “Dense packings of polyhedral: platonic and archimedean solids”, Physical Review, 80, pp. 4-24, 2009.

Torquato, S. & Jiao, Y., “ Robust algorithm to generate a diverse class of dense disordered and ordered sphere packings via linear programming”, Physical Review, 82 (6), 2010.

Van Zeghbroeck, B. J., “Bravais Lattices”, Accessed August 2015, <http://ecee.colorado.edu/~bart/book/contents.htm>

Weller, R. and Zachmann, G., “ProtoSphere: A GPU Assisted Prototype guided sphere packing algorithm for arbitrary objects”, ACM SIGGRAPH ASIA, pp. 1-2, 2010.