Sphere Polygon Interaction to fill Volume Objects

Interacción Esfera Polígono para Rellenar Objetos Volumétricos

Interação Polígono Esfera para Preencher Objetos Volumétricos

Francisco Alejandro MADERA RAMÍREZ[1], Francisco José MOO MENA[2], Jorge Ricardo GÓMEZ MONTALVO[3]

Recibo: 15.02.2017 – Aprobación: 15.01.2020
DOI: https://doi.org/10.30554/ventanainform.40.1909.2019

Artículo de investigación científica y tecnológica4

Resumen: *El cálculo del volumen de un sólido puede aproximarse de manera discreta rellenándolo con primitivas geométricas regulares y convexas. Simular el llenado de volúmenes con esferas en un entorno gráfico tridimensional requiere del movimiento e interacción entre las primitivas geométricas involucradas, en particular la detección y repulsión de las colisiones entre las esferas. El objeto a rellenar se representa con una malla de polígonos y el problema aparece cuando hay muchas esferas moviéndose, las operaciones incrementan y la simulación se vuelve inestable. En este trabajo proponemos un algoritmo para llenar objetos sólidos con esferas, mediante el descarte de primitivas geométricas (esferas, polígonos) para aligerar el proceso. Se muestra formalmente el beneficio de descartar primitivas geométricas sin afectar el proceso de llenado de volumen y se realizan experimentos para*

1 Academic information. Lecturer, Universidad Autónoma de Yucatán, Ciencias de la Computación, Facultad de Matemáticas (Mérida, Yucatán, México). Email: francisco.madera.ramirez@gmail.com. ORCID: https://orcid.org/0000-0002-4495-2017

2 Academic information. Lecturer, Universidad Autónoma de Yucatán, Ciencias de la Computación, Facultad de Matemáticas (Mérida, Yucatán, México). Email: mmena@correo.uady.mx. ORCID: https://orcid.org/0000-0002-8812-2525

3 Academic information. Lecturer, Universidad Autónoma de Yucatán, Ciencias de la Computación, Facultad de Matemáticas (Mérida, Yucatán, México). Email: jgomez@correo.uady.mx. ORCID: https://orcid.org/0000-0002-5606-7517

4 Article from the project *Preservación de volumen en cuerpos articulados*, executed in the period *(02.06.16-11.12.17)*, and enrolled in the research group T*ecnologías Emergentes* de Computación of *Universidad Autónoma de Yucatán.*

determinar el beneficio temporal al usar el algoritmo sugerido en objetos convexos y no-convexos.

Palabras clave: *detección de colisiones, animación de esferas, simulación de partículas.*

Abstract: *Volume computation can be approximated by filling with regular convex geometric primitives. Volume filling simulation using spheres in a 3D environment, requires the motion and interaction between the geometric primitives, specifically collision detection and collision response. The object to be filled is represented by a polygonal mesh, and the problem arises when there are many spheres moving around, operations increase, and the simulation becomes unstable. We propose an algorithm to fill solid objects with spheres, by discarding geometric primitives (spheres and polygons) to speed up the simulation. We emphasize in the benefit of the method, which is shown mathematically and computationally with experiments of convex and non-convex objects.*

Keywords: *collision detection, spheres animation, particle simulation.*

Resumo: *O cálculo do volume de um sólido pode ser aproximado discretamente preenchendo o dito volume com primitivas geométricas regulares e convexas. Simular o preenchimento de volumes com esferas em um ambiente gráfico tridimensional requer o movimento e a interação entre as primitivas geométricas envolvidas, em particular a detecção e a repulsão das colisões entre as esferas. O objeto é representado com uma malha de polígonos e o problema aparece quando há muitas esferas em movimento, as operações aumentam e a simulação se torna instável. Neste artigo propomos um algoritmo para preencher objetos sólidos com esferas, descartando primitivos geométricos (esferas, polígonos) para clarear o processo. O benefício de descartar primitivas geométricas sem afetar o processo de enchimento de volume é formalmente mostrado e experimentos são realizados para determinar o benefício temporal usando o algoritmo sugerido em objetos convexos e não convexos.*

Palavras-chave: *Detecção de colisão, animação de esferas, simulação de partículas*

Introduction

Volume calculation is fundamental in several applications such as engineering, fluids, and optimization problems. An approximation to the volume can be calculated by filling the volume with spheres. The problem can be enunciated as follows: given a volumetric object and appropriate boundary conditions, compute the corresponding set of spheres that fills the object.

The volume filling can be seen as a packing process, where the most computationally costly part is usually the collision detection (HERRERA ZAPATA, 2014). The speed depends critically on how the objects are represented and manipulated. The volume of an object can be approximated as , where n is the number of spheres of radii r that fill the object; spheres are joined and non-overlapped. The method proposed consists on the immobilization of spheres when their motion has stopped, so that spheres have finished moving and they have occupied a permanent location in the object. This immobilization is called the frozen process and indicates that spheres cannot be moved anymore.

The contributions of this work are as follows: An algorithm of the frozen method to discard spheres in the running process is proposed, the mathematical framework is formulated, the analysis of the time and spatial complexity of the sequential version is done, and experiments are conducted to determine the speed up of the simulation.

A class of computer methods exists to generate packing structures by exploring the geometrical constraints. These methods are commonly referred to as packing algorithms. They are designed specifically to generate the structure, rather than simulate the process, of particle packing.

Some projects in the thematic of packing algorithms are as follow:

- Weller & Zachmann (2010) proposed a novel method for filling arbitrary objects very quickly and stably with sets of non-overlapping spheres, such algorithm was able to efficiently compute a space filling sphere packing for arbitrary objects.
- Shimada & Gossard (1995) developed a circle-packing method called bubble mesh to generate triangular meshes for two and three dimensions. Their packing scheme is based on the simulation of the particles that interact with each other under repulsive and attractive forces.
- Voronoi approaches to bound a 3D object with spheres are found in (SHIER, 2013) and (BORKOVEC, 1994).

- A geometric packing generation algorithm was presented in (JERIER, 2009), it is based on a tetrahedral mesh to make an isotropic and dense packing of polydisperse spheres in a short computation time.
- (MADERA et al., 2015) and (MADERA et al., 2013) propose algorithms to fill tubular and non-tubular objects respectively using spheres.

There are some physical simulation models, such as distinct element method (DEM) or molecular dynamics methods, which can take the real interaction forces into account and simulate the dynamic process of particle packing and generate, as a result, the packing structure (JIA, 2001). (PIANET, 2011) compare different ways of using the DEM for consolidating and compressing particle packings in the context of paper-coating applications. (MÜLLER, 2011) connected particles to form a simulation mesh. These particles are represented by anisotropic shapes such as ellipses which are replaced by a sphere tangent. In the field of medicine, the filling of vessels is performed for simulation by inflating balls to treat stenosis, a partial or total blockage of an artery (LUBOZ, 2014). Filling with different 3D shapes can be found in (SHIER, 2013), where Shier and Bourke fill any spatial region with a random fractalization.

Unlike the aforementioned techniques, our method reduces the number of the collision detections and collision responses among spheres and polygons by discarding primitives as the object is being filled. The paper is organized as follows. The method proposed is described in Section 2. Section 3 discusses the sequential implementation. Section 4 gives details of the experiments with the four objects utilized. Finally, conclusion and further work are presented in Section 5.

1. Theoretical foundation

A graphic simulation of the filling volume with spheres is performed. The animation process involves motion, collision detection and collision response. A sphere is represented by center and radius, and contains velocity and position (SCHENEIDER et al., 2003). Spheres are moved by using the Euler numerical method, where the solver calculates the new position and updates the velocity. This new velocity causes the sphere to continue traveling downwards and collides with the polygons of the mesh. The collision response calculates the corrected position and direction of the sphere for the next movement. The fact that spheres are rotationally invariant is most useful.

Having an object mesh of m polygons and assuming it is filling with n spheres, we can enunciate equation (1) which considers the interaction between spheres and polygons.

$$g(n,m) = SS(n) + SP(n,m) \quad (1)$$

SS(n) gives the number of operations of the sphere-sphere interaction, and SP(n,m) returns the number of operations of the sphere-polygon interaction. During the simulation, spheres come down, and collide with the polygons of the mesh, keeping together.

The first term of equation (1), SS(n), can be extended to two factors to indicate the collision detection and the collision response processes (equation 2). The second term of equation (1) equals SP(n,m) = O(nm).

$$SS(n) = O(n^2) + O(n^2) \quad (2)$$

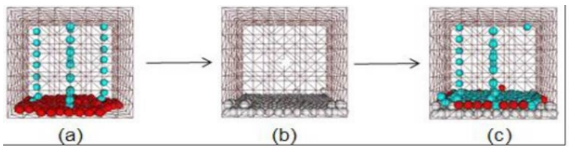
1.1 Frozen Spheres

A group of spheres x is formed by g spheres, then ξ_1= {⊙1, ⊙2,…, ⊙$_\gamma$}, ξ_2= {⊙$_{\gamma+1}$, ⊙$_{\gamma+2}$,…, ⊙$_{2\gamma}$}, leading $\beta = n/\gamma$ the number of groups (CHÍ PÉREZ, 2016). The object is a volume formed by a polygon mesh {△1, △2, …, △m} (SÁNCHEZ UICAB, 2017). The simulation allows appearing spheres by groups, so that ξ_1 starts moving, and when spheres are allocated in the mesh object, the next group appears and starts moving around. The Brute Force approach requires $\frac{(\beta\gamma)^2}{2}$ collision detections as shown in Figure 1B. The number of collision responses equals the double of the number of collision detections since there is a pair of spheres involved.

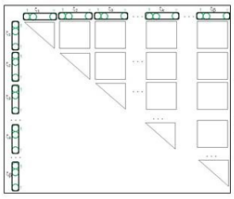
A sphere ⊙ is defined by their coordinates ⊙.x, ⊙.y, ⊙.z, and it can be converted into a frozen sphere ○. During the process, running spheres ⊙ are tested against the other spheres. The Frozen approach consists on applying the immobilization to spheres that have been allocated in the object mesh. Frozen spheres are placed on the bottom part of the polygonal mesh, due to the gravity, and they have no motion. Frozen spheres are also considered for collision detection tests; however, they are not considered for collision response due to their immobility (Figure 1A).

The first term of equation (2) relates to the number of collision tests. The number of tests required for self-collisions of a group equals $\frac{\gamma^2}{2}$ and it is represented by triangles of the main diagonal (Figure 1B). The number of collision tests for spheres, of different groups, is represented by the squares placed above the main diagonal and gives γ^2 each, ξ_i vs ξ_j. From here, spheres of group 1 must test collisions with spheres of groups 2, …, β, giving $(\beta\text{-}1)\,\gamma^2$ tests. This way, $(\beta\text{-}1)\,\gamma^2 + \frac{\gamma^2}{2}$ tests are needed for group 1, $(\beta\text{-}1)\,\gamma^2 + \frac{\gamma^2}{2}$ tests for group 2, and so on. Accordingly, for β groups, $\beta\frac{\gamma^2}{2} + \gamma^2 \sum_{k=1}^{\beta-1} k = \beta^2\frac{\gamma^2}{2}$ tests are required.

If a collision of a pair of spheres occurs, then both spheres must move, in opposite directions, for response. As a result, twice the number of operations is obtained for collision detection: $2\beta^2\frac{\gamma^2}{2}$. Therefore, $SS(n) = SS(\beta\gamma) = \beta^2\frac{\gamma^2}{2} + \beta^2\gamma^2 = \frac{3}{2}(\beta\gamma)^2$. As the Frozen method immobilizes some spheres, a benefit in the processing is achieved; thus, lemma 1 is stated to show the reduction in the number of operations by applying the frozen method in spheres groups.



A The filling process (a) blue spheres are coming down, red spheres collide with polygons, (b) white spheres are frozen, (c) the process continues after the previous group has been frozen



B Sphere-Sphere interaction using groups, the brute force approach

C (a) Frozen spheres in white, (b) discarded spheres in yellow, (c) running spheres in blue

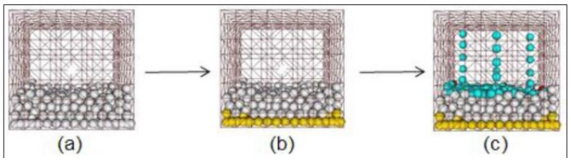


Figure 1. *(A) The volume filled with spheres, (B) the interaction between spheres, (C) the frozen spheres are discarded.*

- Lemma 1. The number of operations of $SS(n) = SS(\beta\gamma)$ with β groups of γ spheres each is reduced from $\frac{3}{2}(\beta\gamma)^2$ to $2\beta - \frac{1}{2}\gamma^2$.

Frozen spheres can be discarded, from $\oslash$ to$\otimes$, which means that they are not considered for collision detection either for a response, they are practically removed from the simulation. Frozen spheres can be discarded when they are located below the running spheres and they are not colliding with them (Figure 1C).

The heights of the running and frozen spheres are compared as follows: $\odot.y - \oslash.y > \varepsilon$. The problem with this constraint is that running spheres are often moving. The heights of frozen spheres are compared instead, the last frozen sphere group against the other frozen spheres:

$$\oslash j.y - \oslash i.y > \varepsilon, \quad \forall - \oslash i.y \in \xi i, \quad \oslash j.y \in \beta_{LFG} \tag{3}$$

Where β_{LFG} is the last frozen group (LFG). This way the number of operations is reduced as stated in lemma 1 to $2(\beta - \frac{1}{2} - \alpha)\gamma^2$, where α is the number of sphere groups discarded.

1.2 Discarded Polygons

At the start, the first group of spheres is running and mg comparisons with polygons are performed (Figure 2A). A polygon $\triangle$ can be discarded when one or more spheres are placed on the polygons, that is, the polygon is covered with spheres in such a way that the following sphere groups will never collide with that polygon, but with covered spheres (Figure 2B).

The number of discarded polygons varies depending on the sphere motion, so that when running ξ_1, then m1 polygons could be discarded; when running ξ_2, then m2 polygons could be discarded, and so on. The number of operations of SP(n,m) with γ spheres per groups is reduced from nm = $\beta\gamma$ to $\beta\gamma$ (m-m1-…-mβ), where mi is the number of discarded polygons in group i.

1.3 Set of Groups

A real simulation demands the motion of several sphere groups at the same time. This analysis is extended to run several groups. Let k be the number of groups to run at the same time, which we call a set of groups. The first k groups run, then, these groups are frozen and the next k groups start running. Assuming k=3 (Figure 2C), R_1 (in green) involves the spheres of the third set in interaction and R_2 (in blue) contains spheres of the third set against spheres of sets 1, 2.

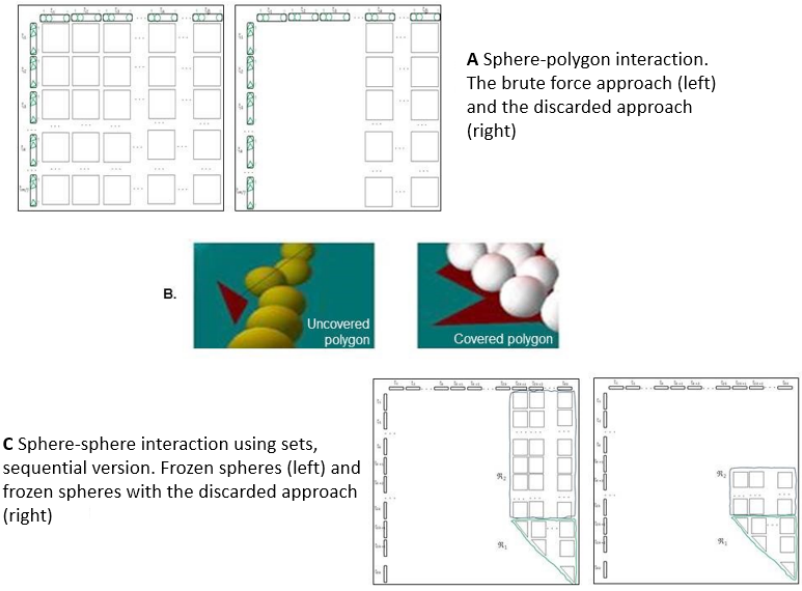


Figure 2. (A) sphere-polygon interaction, (B) a covered polygon, (C) sphere-sphere interaction.

In $\mathbb{R}_1$, a collision response is performed in each sphere of the pair collided, so that it is the double of the number of colliding pairs. However, in $\mathbb{R}_2$ only the half of the number of spheres are considered, due to spheres of sets 1, 2 are frozen. $\mathbb{R}_2$ involves the interaction between spheres of set 3 and spheres of sets 1, 2, requiring $(k\gamma)^2+(k\gamma)^2$ collision tests. $\mathbb{R}_1$ involves the interaction between spheres of set 3, requiring $\frac{(k\gamma)^2}{2}$ collision tests. The number of collision responses needed is $(k\gamma)^2+(k\gamma)^2 + 2\frac{(k\gamma)^2}{2}$ as illustrated in Figure 1B left. Thus, lemma *2* is enunciated.

- Lemma 2. The number of operations of $SS(n)=SS(\beta, \gamma, k)$ with β groups of γ spheres each with a set of k groups equals $(\beta\text{-}k)\ (k\gamma)^2+\frac{(k\gamma)^2}{2}+(\beta\text{-}k)\ (k\gamma)^2+2\frac{(k\gamma)^2}{2}$.

Evidently discarding spheres reduces the number of spheres to deal with. As a result, lemma 2 is transformed in $(\beta\text{-}k\text{-}\Sigma\alpha i')\ (k\gamma)^2+\frac{(k\gamma)^2}{2}+(\beta\text{-}k\text{-}\Sigma\alpha i')\ (k\gamma)^2+2\frac{(k\gamma)^2}{2}$, where α_i' indicates if the group i is discarded, $0\leq i\leq\beta$. Hence, $\alpha i'=1$ if the group i is discarded, otherwise $\alpha_i'=0$. Alike discarding spheres, discarded polygons are independent of the spheres sets. Polygons are also discarded, and lemma 3 is enunciated.

- Lemma 3. The number of operations of $SP(n,m)$ with γ spheres per group each and a set of k groups is $k\beta(m\text{-}m1'+ m2'+\ldots m\frac{\beta}{k'})$, where $m1'= m1+ m2+\ldots mk$ and $m2'= mk+1+ mk+2+\ldots m2k$, etc.

Therefore, less polygons are employed in every frozen activation and the ideal case would be that this sum equals m when the object is filled. Alike lemma 2, the a value is included to consider frozen and discarded spheres to reduce the number of operations stated in lemma 3.

2. Methodology

2.1 The Sequential Algorithm

Using sphere groups, the number of operations of the frozen approach is shown in equation (4) and applying the discarded process in spheres and polygons, $\beta=\beta-\alpha1\text{-}\alpha2\text{-}\ldots\text{-}\alpha\beta$ and $m=m\text{-}m_1\text{-}m_2\text{-}\ldots\text{-}m\beta$ are set.

$$SS(\beta\gamma) + SP(\beta\gamma,m) = \frac{3}{2}(\beta\gamma)2+2(\beta\gamma)2m \quad (4)$$

Using spheres sets, the number of operations required for the frozen method without and with the primitives discarded is shown in equation (5) and applying the discarding process in spheres and polygons, $\beta=\beta\text{-}\alpha'1\text{-}\alpha'2\text{-}\ldots\text{-}\alpha'\frac{\beta}{k}\ m=m'\text{-}m'1\text{-}m'2\text{-}\ldots\text{-}m\frac{\beta}{k}$ are set.

$$SS(\beta\gamma) + SP(\beta\gamma,m) = \quad (5)$$
$$(\beta\text{-}k)\ (k\gamma)^2+\frac{1}{2}(k\gamma)^2+(\beta\text{-}k)\ (k\gamma)2+(k\gamma)^2+\frac{\beta}{k}(k\gamma)m$$

Algorithm 1 shows the pseudocode of the sequential algorithm. There are three loops to detect collisions. The first cycle refers to the SS interaction of running spheres, the second cycle relates to the SS interaction of running and frozen spheres, and the third cycle refers to the SP interaction. The number of groups currently running are start, start+1, …, end. The FFG (First Frozen Group) is the following group to the last group discarded; if no discarded groups exist, then FFG=0. The LFG (Last Frozen Group) is the last group frozen, and the following groups are currently running.

There are two procedures to be included, the discarded spheres and the discarded polygons. These procedures are not shown in algorithm 1, since they are not called every time step such as the collision detection and the collision response operations. To discard spheres, a comparison between the LFG group and the other frozen groups is required, considering the constraint of equation (3). In the case of polygons, the discarding process demands the verification of the covering spheres. Colliding spheres can be considered as a covered sphere; however, the polygon can be larger that several spheres are needed. A ray sphere intersection test is performed for each of the vertices of the polygon. A vertex has a normal vector that works as a ray; therefore, the three vertex arrays must collide with a sphere to be considered as a discarded polygon. The ray sphere intersection test considers the verification of the closeness of the sphere and the polygon.

Table 1. The Collision Detection Algorithm.

Algorithm 1: Spheres Collision

```
For each (Oi, Oj,) start ≤ i,j ≤ end
       If Collision (Oi, Oj,) { response(Oi), response(Oj) }

For each (Oi, ◇j,) start ≤ i ≤ end, FFS ≤ j ≤ LFS
     If Collision (Oi, ◇j,) { response(Oi)}

For each (Oi, △j,) start ≤ i ≤ end, 0 ≤ j ≤ NumGroupPolys
     If Collision (Oi, △j,) { response(Oi)}
```

3. Results and discussion

3.1 Description of results

Algorithms were implemented on a PC Intel Xeon two-CPU 2.49Hz with a GeForce 590 GTX Graphics Card. Four objects were employed: Torus

(1,1152 polygons), Sphere (960 polygons), Cube (768 polygons), Human (2,218 polygons). The time of the spheres and polygons interaction during the running time is analyzed. Spheres are created in blocks; the algorithm finds the best location of the spheres to fill the volume and frozen polygons are counted.

The animation process can be implemented according to several parameters such as velocity and collisions. To evaluate the algorithm performance, the number of frames per second is tested. Several sphere groups appear inside the object, the time is recorded, and spheres are frozen.

The Torus and Sphere objects have a 10-sized group while the Cube and Human have 3-sized group. The time was recorded in every set: two groups for the Torus, Sphere, and Human and seven groups for the Cube. The frozen process and the discarding process are activated every two sets in the Torus, Sphere and Cube, and every three sets in the Human.

The time considered is the sphere-polygon interaction due to the sphere-sphere interaction is irrelevant (0-1 ms). It means that the comparison between spheres does not affect the algorithm performance. The Torus object is filled with 320 spheres. Two sets are leaving to come down before the frozen process. The first set varies in the range of 13-25 ms and the second set varies in the range of 26-50 ms. 51% of discarded polygons are reported.

In the Sphere object, 23 ms and 35 ms are obtained for the first and second set, respectively. The time is reduced due to the object provides plenty of space to allocate spheres. Few numbers of discarded polygons are obtained, 0.4%. In the third object, the Cube, the first set takes 17 ms and the second set takes 35 ms. 0.8% of discarded polygons are achieved. In the Human object the time recorded was 13, 26, and 39 ms for the sets i, i+1, i+2, respectively. There was 0.6% of discarded polygons.

Since the slower processing time a linked list to trace polygons is constructed rather than using a sequential loop. The linked list connects the polygons sorted, considering the discarded polygons. If $\triangle_i \rightarrow \triangle_j \rightarrow \triangle_k$, and $\triangle_j$ is discarded, then $\triangle_i \rightarrow \triangle_k$.

To speed the implementation up, a Quad Tree is added in the mesh object. The mesh is divided into four parts and a sphere is tested with the quarter part of the polygons. This accelerates the algorithm as shown in Figure 3. The Quad Tree speeds up the algorithm, but there is a limited number of animating spheres per set: 40 spheres in the Torus, Sphere and Cube, and 100 spheres in the Human. For real time simulations, a large number of sets is required.

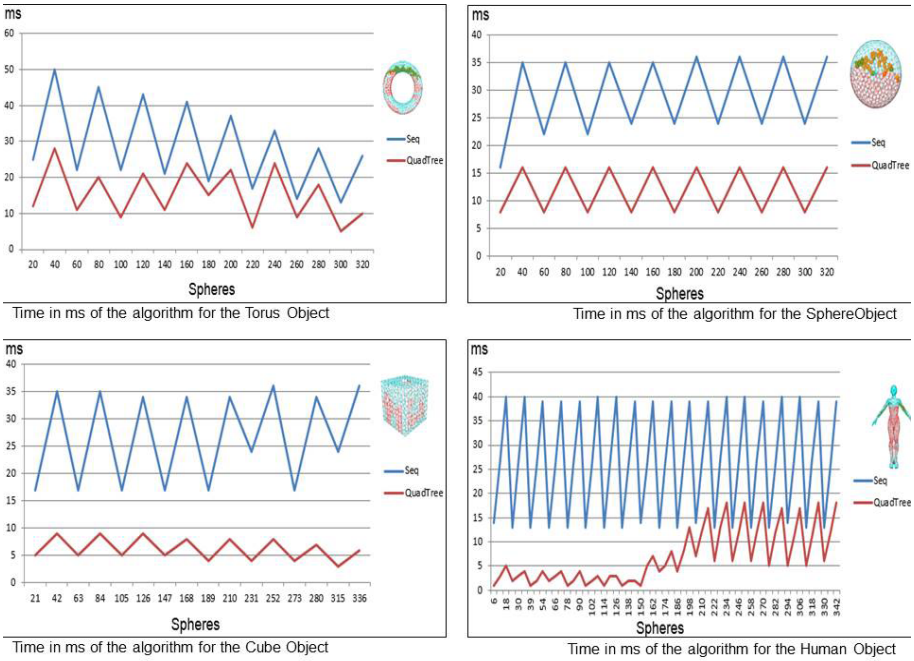


Figure 3. *Algorithm time in ms for the Torus, Sphere, Cube and Human objects.*

3.2 Discussion of results

Convex objects are easier to fill and shows a time processing similar during the whole simulation, from 20 ms to 35 ms. Non-convex objects are more difficult to be filled but the time decreases as the simulation evolves. Experiments show a better time when using the algorithm with a Quad Tree, in particular convex objects (sphere and cube) have the best improvement, about 10 ms per frame better than the sequential version. The Quad Tree demonstrates benefits in non-convex objects, but the enhancement is lesser, from 10 ms to 5 ms better. The reason is that a non-convex object presents concavities, where spheres should be accommodated, but takes longer and it its more difficult to obtain their final position.

The frozen process guarantees the reduction in the number of spheres and polygons considered; in such a way that the number of geometric primitives is being lesser as the simulation runs.

3.3 Comparison with other approaches

Spheres (particles) has been used for physical simulations during several years. The methods are being improved and new technology helps

to run simulations in real time. The methods integrate the velocities and positions of all particles for a small interval of time. In this section, recent approaches are described, emphasizing in the similarities and differences with our proposal method.

Kalms (KALMZ, 2019) employed 2D spheres in a parallel implementation, but he did not use an object volume, just spheres moving around. His algorithm takes n log n time running on GPU using CUDA, and particles are grouped in a hierarchy, similar to our approach where a tree data structure is used to speed up the simulation.

Spheres have been used to model several types of physical phenomena. Macklin et al. (MACKLIN, 2014) utilized particles to model gases, liquids, deformable solids, rigid bodies. They employed position-based dynamic method to deal with particles, which contain physical parameters such as friction, stiff components, and others. Similar to ours, the collision detection method computes the distance between the spheres, and they address the positional drift by freezing particles in place if their velocity has dropped below a user-defined threshold.

In (XU, 2019), the interaction forces between different particles and between particle and boundaries are described by hard spring repulsion or a decaying repulsion potential that are introduced to prevent collisions between particles. The number of 3D particles simulated are few, about 100 or 200, but authors applied magnetic field intensity on the time response of MRF (magnetorheological fluid).

The study of Fukuda and Fukuoka (FUKUDA, 2019) involved an examination of the effects of particle shapes and the coefficients of contact forces used in the discrete element method on mixture flows. Different flows were obtained for spherical and gravel particles (joined spheres) under dense conditions, whereby the particle shear stress became much greater than that for water. The repulsion process is similar to ours, by using the normal vectors to cull away spheres.

Melero et al. (MELERO, 2019) introduced the EBP-Octree, a tight hierarchy of convex bounding volumes, so it can be included in the traditional approach for pairwise collision detection of rigid models. The method quickly rejects non-intersecting objects. They do not offer just basic first contact detection, but also intersection phase with the exact list of collided triangles. They ran simulations with massive models, 28 millions of 3D spheres.

Yan et al. (YAN, 2019) applied the recently developed updated Lagrangian particle hydrodynamics (ULPH) method to model and simulate weakly compressible multiphase flows using 2D spheres. Several physical equations were considered for fluids.

As can be seen, the running simulations employed different hardware such as GPUs. All approaches include physical properties, which achieve a real simulation, but requires more processing. The main phenome to simulate is the fluid and its variations. Some approaches simulate the motion of few spheres; only one approach simulates millions of spheres. One approach employs similar computation for collision detection and other approach for collision response. In addition, one approach utilizes a tree structure to cull away primitives, this save time in the running time, but requires a preprocessing stage to prepare the Bounding Volume.

4. Conclusion

A method which employs spheres to fill object volumes is proposed. The method provides the frozen process, which allows discarding spheres and polygons to reduce the number of operations during the run time. The difference with other approaches lies on the discard of primitives due to the frozen process, which allows avoiding expensive computations. The number of spheres increases when the volume is filling, but the number of operations holds. The method can be adjusted to a specific animation to simulate fluids, solids, smoke, etc. This method can be speeded up with a type of structure such as the Quad Tree employed in the experiments.

As further work, the algorithm is suitable to be parallelized by considering the sphere groups as handling with a thread. In addition, the animation can be extended to allow the motion of the volumetric object, making spheres to be unfrozen.

Bibliographic references

BORKOVEC, M.; DE PARIS, W; PEIKERT, R. (1994). The fractal dimension of the apollonian sphere packing. Fractals, vol. 02 No. 04, pp. 521-526. ISSN: 1793-6543.

CHI PÉREZ, MARTÍN LEONEL. (2016). Paralelización de detección de colisiones en humanos virtuales. Tesis de grado (Maestría en Ciencias de la Computación). Mérida, México. Universidad Autónoma de Yucatán, Facultad de Matemáticas. Clasificación Biblioteca en código de barras: T0004185 INGE

FUKUDA, Tomoo; FUKUOKA, Shoji (2019). Interface-resolved large eddy simulations of hyper concentrated flows using spheres and gravel particles. In Advances in Water Resources, Volume 129, Pages 297-310, ISSN 0309-1708.

HERRERA ZAPATA, CARLOS GASPAR. (2014). Algoritmos en Paralelo para preparación de objetos para detección de colisiones. Tesis de grado (Maestría en Ciencias de la Computación). Mérida, México. Universidad Autónoma de Yucatán, Facultad de Matemáticas. Clasificación local en código de barras: T0004051 INGE

JERIER, Jean-Franois; IMBAULT, Didier; DONZE, Frederic-Victor; DOREMUS, Pierre (2009). A geometric algorithm base on tetrahedral meshes to generate a dense polydisperse sphere packing. Granular Matter, vol. 11, No. 1, pp. 43-52. ISSN: 1434-7636.

JIA, Xiaodong. & WILLIAMS, Richard A. (2001). A packing algorithm for particles of arbitrary shapes Powder Technology, Vol. 120, No. 3 (oct), pp. 175 - 186. ISSN: 0032-5910.

KALMS, Mikael (2015). "High-performance particle simulation using CUDA". Master Thesis, Department of Electrical Engineering, Linköping University. June 3, 2015.
http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A816727&dswid=8419

LUBOZ, Vincent; KYAW-TUN, Jim; SEN, Sayan; KNEEBONE, Roger; DICKINSON, Robert; KITNEY; Richard; BELLO, Fernando (2014). Real-time stent ad ballon simulation for stenosis treatment. In Visual Computing, vol. 30, No. 3, pp. 341-349. ISSN: 0178-2789.

MACKLIN, Miles; MüLLER Matthias; CHENTANEZ, Nuttapong; KIM, Tae-Yong (2014). Unified particle physics for real-time applications. ACM Trans. Graph. 33, 4, Article 153 (July 2014), 12 pages 153:1-12, DOI: https://doi.org/10.1145/2601097.2601152

MADERA, Francisco A.; AYALA, Enrique; MOO-MENA, Francisco (2015). In Proceedings of the 10th International conference on Computer Graphics and Applications (VISIGRAP 2015), pp. 325-331. http://www.visapp.visigrapp.org/?y=2015

MADERA, Francisco A.; LAYCOCK, Stephen D.; HERRERA, Carlos (2013). In Proceedings of the IASTED International conference on Computer Graphics and Imaging (CGIM 2013), pp. 70-76.

MELERO, Francisco Javier; AGUILERA, Ángel; FEITO, Francisco Ramón (2019). Fast collision detection between high-resolution polygonal models. In Computers & Graphics, Volume 83, Pages 97-106, ISSN 0097-8493.
http://www.actapress.com/PaperInfo.aspx?paperId=454993

MÜLLER, Matthias; CHENTANEZ, Nuttapong (2011). Adding physics to animated characters with oriented particles. In Virtual reality Interactions and Physical Simulations (VRI-Phys), pp. 83-91.

PIANET, G.; BERTRAND, F.; VIDAL, D.; MALLET, B. (2011). Discrete element method-based models for the consolidation of particle packings in paper-coating applications. Asia-Pacific Journal of Chemical Engineering. Vol. 6, No. 1, pp. 44-54. ISSN: 1932-2143.

SÁNCHEZ UICAB, GONZALO AUGUSTO. (2017). Preservación de volumen en cuerpos articulados. Tesis de grado (Maestría en Ciencias de la Computación). Mérida, México. Universidad Autónoma de Yucatán, Facultad de Matemáticas. Clasificación local en código de barras: T0004409 INGE

SHIER, John & BOURKE, Paul (2013). An algorithm for random fractal filling of space [online]. In: Computer Graphics Forum, Vol .32 No. 8, p. 89-97. ISSN : 1467-8659.

SHIMADA, Kenji & GOSSARD, David C. (1995). Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing. Third ACM Symposium on Solid Modeling and Applications, SMA'95 (17-19/05/1995). Salt Lake City (UT, USA): ACM. HOFFMAN, Chris & ROSSIGNAC, Jarek (eds.). Proceedings of the SMA'95. New York (NY, USA): Association for Computing Machinery, ACM. p. 409–419. ISBN: 0-89791-672-7.

SCHNEIDER, PHILIP; EBERLY, DAVID H. (2003). Geometric Tools for Computer Graphics. San Francisco, USA. ISBN: 1-22860-594-0. The Morgan Kaufmann series in Computer Graphics and Geometric Modeling.

WELLER, Rene & ZACHMANN, Gabriel (2010). ProtoSphere: a GPU-assisted prototype guided sphere packing algorithm for arbitrary objects. ACM SIGGraph Asia 2010 Sketches, SA'10 (15-18/12/2010). Seoul (Republic of Korea): ACM. CANI, Marie-Paule & SHEFFER, Alla (eds.).Proceedings of the SA'10. New York (NY, USA): Association for Computing Machinery, ACM. p. 8:1–8:2. ISBN: 978-1-4503-0523-5.

XU, Jinhuan; LI, Jianyong; ZHU, Pengzhe; LI, Baozhen; ZHAO, Chaoyue (2019). Coarse-grained molecular dynamics simulations of particle behaviors in magnetorheological polishing fluid. In Computational Materials Science, Volume 163, Pages 68-81, ISSN 0927-0256.

YAN, Jiale; LI, Shaofan; ZHANG, A-Man; KAN, Xingyu; SUN, Peng-Nan (2019). Updated Lagrangian Particle Hydrodynamics (ULPH) modeling and simulation of multiphase flows. Journal of Computational Physics, Volume 393, Pages 406-437, ISSN 0021-9991.