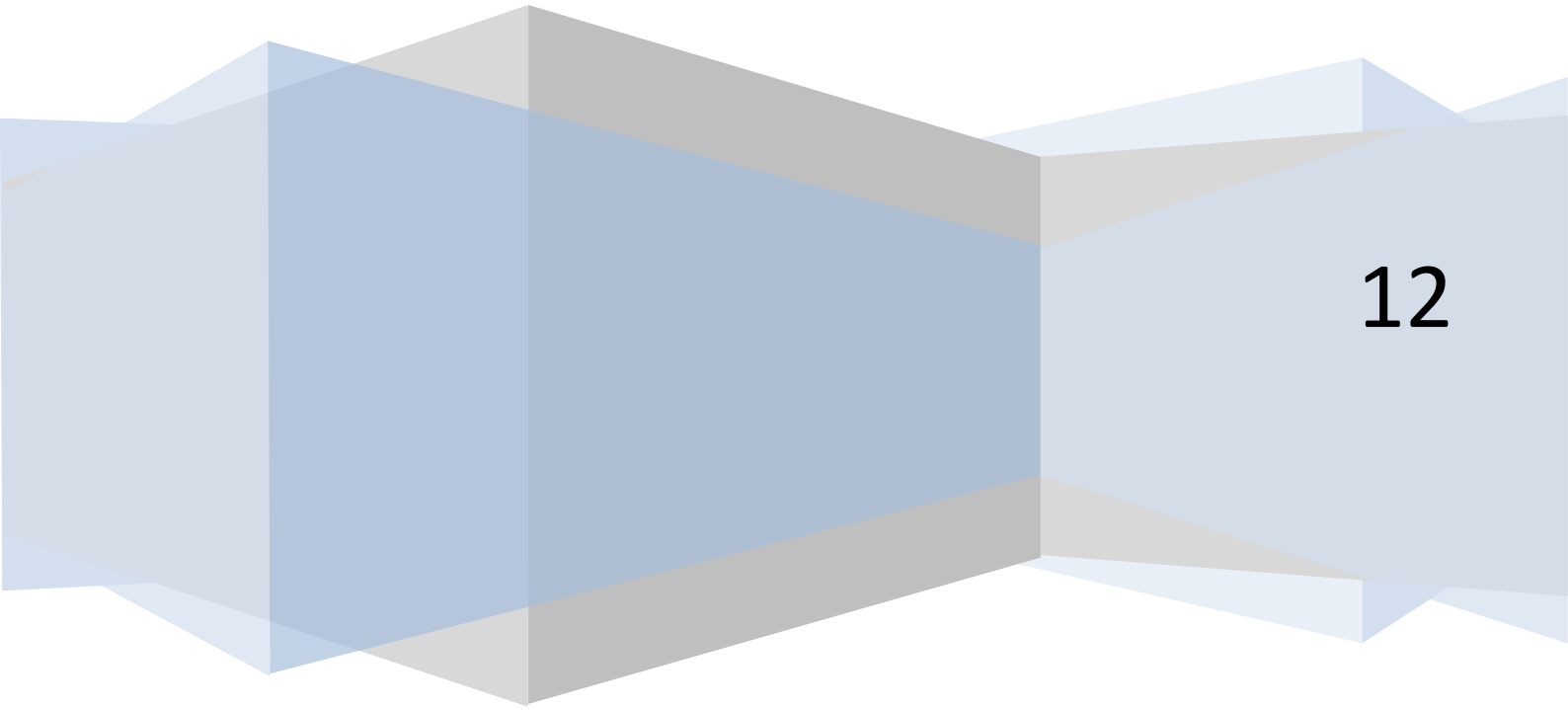
Bournemouth University

Sand Simulation

Athanasios Papadimitriou

Msc Computer Animation and Visual Effects



12

ABSTRACT

The simulation of granular materials has been considered a changeling task in computer graphics world. Given the limitations that a granular system presents, these materials appear unique properties that sometimes are difficult to be simulated with physical accuracy. Sand is a granular material that has offered some great effects in computer animation. Nevertheless, it is an expensive simulation as a ton of particles is needed to interact with each other in order to create visually interesting results. In this master thesis, a hybrid approach for sand simulation is proposed combing the use of particles and a heightfield for the creation of sand surfaces and the simulation of its properties. This hybrid approach is succeeded by taking advantage of the sand as an opaque material and the replacement of the invisible part of a sand surface by a mesh that is defined by a heightfield.

Table of Contents

1. INTRODUCTION

In recent years there has been a significant increase of interest in simulating the dynamic motion of granular materials, such as sand, soil, powders and grains. In computer graphics world there has been always the demand of animating such materials, but their unique natural properties has made this task a challenging procedure. Especially, sand is a granular material that is met in many landscapes and its interaction with the environment has been researched for many years. Animating scenes with sand in computer graphics has become an interesting topic of research as the computational models for this should be highly configurable with rich visual behaviour and in as low as possible computational cost.

Many techniques have been developed for simulating sand either as a solid material or as a fluid. Due to the physical properties of the sand, sand can flow under external forces as a fluid and stabilize into sand piles as a solid. Animating sand a fluid is a nice way to lower the computational cost of sand animation but it lack in terms of physical accuracy, as the solid characteristics of sand grains are not implemented. If we want to simulate a physical model of sand movement, each grain of sand will be considered as a discrete element. So a particle-based approach in sand simulation is required for succeeding physical accuracy. The main disadvantage of such an approach is the high computational cost of having such a great number of particle's interaction. Many computational models have been developed to represent the properties of sand. Discrete element methods (DEM) are popular in this area as they can be applied in each grain and produce realistic results in terms of physical accuracy. But the computational efficiency of these methods is still a challenge.

Another way of simulating sand surfaces suggests to animate these kind of surfaces as a continuous volume of material. The most common model is the use of a heightfield which is created by dividing the volume into a uniform rectilinear grid. The use of such a model offers the ability to animate any displacement in a sand surface and handle many types of collision between an object and the surface. But even if these methods are efficient in terms of computational cost, they only can be used in animating sand surfaces and what happens on it. Other natural phenomena including sand are not able to be implemented without the use of a particle system and the representation of a sand grain as a discrete element.

This master thesis involves the implementation of a hybrid technique for sand simulation. By taking advantage of the fact that sand is an opaque material, a sand surface could be divided into two layers: the visible surface layer and the invisible interior layer. In the surface layer, sand simulation is implemented by using a particle system with Discrete Element Method in order to

apply the physical properties of the sand. For the invisible layer, a heightfield is used that stores in a 2D matrix the height of each part of the surface forming vertical columns. So, each part of the surface will have a predefined height with a surface of particles at the end of each column. While a simulation consists of the simulation itself and a visualisation part, the scope of this project has been limited to the simulation mainly.

In the following chapters the elements and the methods that were used for this sand simulation are discussed. In chapter 2, an overview of the previous work in the field of the simulation of granular materials is presented. In chapter 3, the main algorithm for the creation of the forces among the particles is explained. In chapter 4, the ground granular model for the representation of a sand surface is analyzed and algorithms for the deformation of the surface are also included. In chapter 5, the basic aspects for the collision tests between all the elements in the simulation are examined. In chapter 6, the simulation of this project and its main elements are discussed. Finally, in chapter 7, the advantages and disadvantages of this simulation are presented with suggestions for future work and improvement of this technique.

2. PREVIOUS WORK

A lot of research has been taken place for the simulation of granular materials, such as sand. Many different techniques have been proposed throughout the years using different elements for the representation of sand and its properties. The previous work in this field can be divided into particle-based methods and mesh-based methods.

Bell et al. (2005) presented a simulation method of granural materials, such as sand and grains, based on theoretical and experimental results in physics area. In their approach, they model the granular material with discrete elements represented by particles. The major advantage of using discrete elements is that dynamic phenomena, such as splashes and avalanches, can be generated by the particles' interaction with great physical accuracy. In addition, the simulation technique that they proposed is based on Molecular Dynamics methods, which include methods for handling the contact forces efficiently. The discrete elements in this technique are represented by multiple round particles that move together as a single rigid object. As a result, they can handle the collisions with other objects in a rigid body simulation system with realistic results. But the high computational cost is a disadvantage, as despite the realistic simulation, a sand simulation can only be used offline. In contrast to the above rigid body approach, Zhu and Bridson (2005) combined particle-in-cell(PIC) and fluid-implicit-particle(FLIP) methods to simulate sand as fluid. The results were computationally more efficient but the motion of the sand as fluid does not match all the natural properties of granular materials in terms of physical accuracy.

Based on previous similar methods, Alduan et al. (2009) presented an improved technique for simulating granular materials with particles that achieves high visual resolution and at a lower computational cost than earlier techniques. Their main difference is detected on the simulation of the internal and external forces of granular materials. The computationally expensive internal granular forces are simulated at a spatial low resolution scale, while the less expensive external forces are simulated at a spatial high resolution scale. So the physical accuracy is preserved in a more efficient way in terms of memory and computation cost. But still the problem of the high computational cost of the internal forces remains, even in a smaller scale, due to the required small time step.

Zhu and Yang (2010) described a new method for simulating sand by separating the forming sand piles in two layers: the static and the moving layer. This method offers a lower computational

cost than other particle-based methods as it replaces the static and invisible particles of sand with a heightfield. Thus, the number of particles is reduced and sand is animated as a surface flow according to the BCRE model. Their results show that their approach performs pretty well in large 3D scenes. But it cannot be used in interactive applications because of the use of the discrete element method for the interaction between the particles in the surface flowing layer.

As far as the mesh-based methods are concerned, Sumner et al. (1999) proposed a method for animating ground surfaces consisted of granular materials. For computational efficiency the ground material is modelled as a heightfield which is formed by vertical columns. So after the impact of a solid object, deformations are applied to the ground surface through the compression of the material and the movement of the material between the columns. Algorithms for the collision detection, the displacement of the surface and the process of the erosion are also presented in order to succeed as realistic results as possible. The biggest advantage of this technique is the low computational cost and its usability in simulations that run in real time. The technique that is described gives the ability to capture many behaviours of substances such as sand, mud and snow, but it lacks when it is tried to animate specific characteristics of the sand, such as the creation of sand piles and the effect of friction when a collision happens.

An improved version of an interactive system with a ground surface composed of a granular material is described by Onoue and Nishita (2003). The main difference from the above technique is that this proposed method can handle objects of various shapes, even concave ones. This enriches the interactivity of the system and the visual results become more realistic. This technique is also based on the use of a heightfield for the representation of the ground surface. For the representation of the granular material on the surface of an object, a height span map is used. Its use creates a nice effect in the interactivity between the ground surface and the different objects. Overall, it is a very interesting approach in simulating granular materials at an interactive frame rate.

Benes et al. (2006) presented a novel approach for a virtual granular material interactive manipulation system. The representation of the sand surface relies on the fact that a sand surface cannot form concave structures and so it is accomplished with a regular heightfield. The displacement and the erosion of the sand surface are considered for the sand manipulation and the results are not so realistic, but they are close to real-time speed. This is also the disadvantage of this system, as several aspects of sand simulation cannot be produced by using only a heightfield. In addition, the collision with non-spherical objects is still an unsolved problem in their method. But the ground surface manipulations are very helpful for creating similar results with low computational cost in an improved version of this technique.

3. THE DISCRETE ELEMENT METHOD (DEM)

Sand is a granular material that is consisted of many discrete solid particles. The sand grains are represented as spherical particles that interact with each other. Two important ingredients for such a granular system is the repulsion between the particles in contact and the dissipation of energy during the collisions. Another important characteristic of the behaviour of sand grains is the fact that when a significant number of sand particles are at rest, they behave as solid. Part of the reason why they behave as solid is the static friction that is taking place when two or more particles are in contact. The static friction is also responsible for the formation of sand piles and their solid situation.

The Discrete Element Method is an approach to numerical simulation that computes the motion of a large number of particles from the individual motion of each particle and their mutual interactions. In this thesis' model, a DEM method is used to calculate the forces between the particles and the motion of each one individually. An individual sand grain is modelled by a spherical particle.

3.1 The DEM algorithm

The basic steps in the algorithm using the Discrete Element Method (DEM) are:

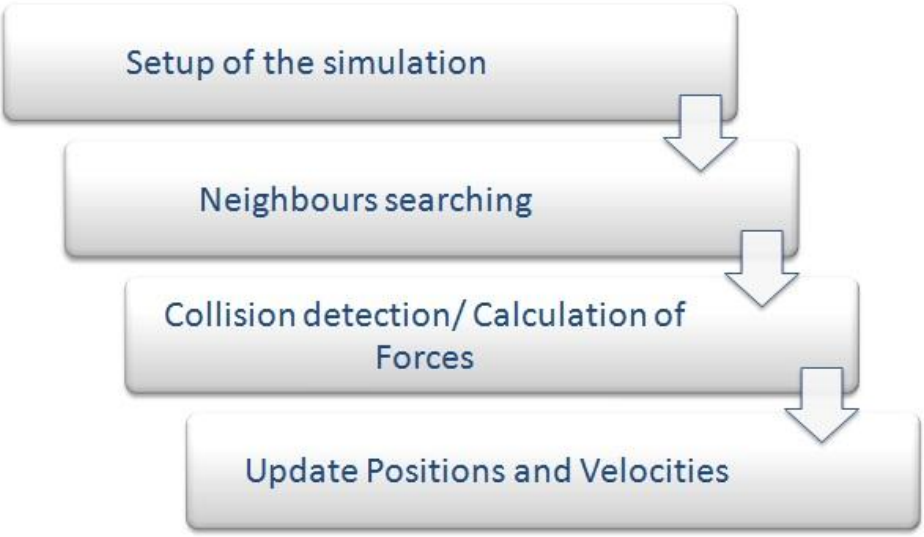


Figure 1. DEM Algorithm

1. Setup: The first step of the algorithm is the generation of particles and arranging them in the environment.

2. Neighbours: After the initialization of the particles in the system, the neighbours of each particle is computed in order to est for collisions. This is a significant part of the algorithm as it determines the particles that are candidates for interaction with each other.

3. Collisions and Calculation of contact Forces: When a collision detection is occurred between two particles, the appropriate forces are applied according to the DEM model that it is described below.

4. Update Positions and Velocities: The total force that is applied to each particle after the collision is incorporated into the motion equations and the acceleration is updated. Therefore, the positions of each particle need to be changed accordingly as well as their velocities.

3.2 Contact Model

In the DEM algorithm, when a collision between two particles occurs, a numerical contact model is applied for the proper simulation of the motion of these particles. This contact model calculates the total force that affects the trajectory of the particle. The total force on a given particle is computed as the sum of its pairwise interactions. The contact model that is used in this simulation is based on the implementation of Lee and Herrman (1992).

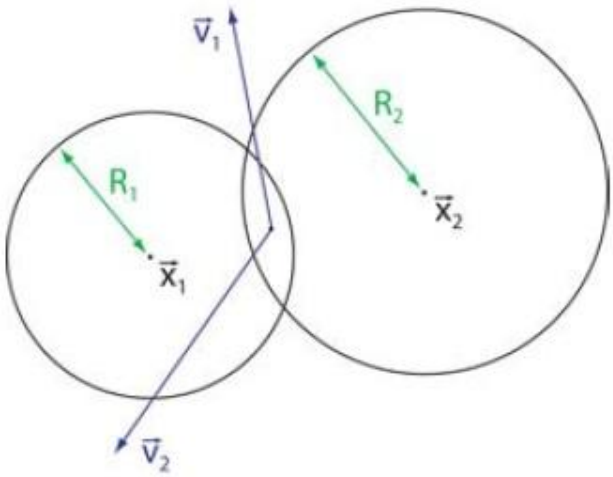


Figure 2. Contact Model

The total force that is responsible to update the acceleration of each particle is the sum of the external and the internal forces. The external forces that act in a particle are the gravity and any other force from the environment. In the external forces, the force from the collision between the particle and an object can be concluded, as well as the force from the interaction between the particle and the boundary wall. The internal forces are the forces from the contacts between the particles. So the total force is compute as:

$$TotalForce = ExternalForces + InternalForces$$

The internal contact force can be divided into normal and tangential components as follow:

$$\vec{F} = \vec{F}_n + \vec{F}_t$$

Consider two particles *i* and *j* that are in contact and their positions are p_i and p_j respectively. Their distance is defined as $r = p_i - p_j$. A vector $\vec{n}$ is also defined to be a unit vector parallel to the distance between the particles *i* and *j*. A vector $\vec{s}$ is orthogonal to $\vec{n}$ and defines the tangential component of the internal force. Consider also the relative velocity $\vec{v}$ as $\vec{v}_i$ - $\vec{v}_j$ and the radius of particle *i* as a_i. So, according to the model of Lee and Herrman, the internal force on particle *i* exerted by particle *j* is :

$$F_{j \to i} = F_n n + F_s s$$

As it is mentioned before, the internal contact force is consisted by two components. Here, F_n if the normal factor of the normal component $\vec{n}$ and F_s is the shear factor of the tangential component s. These two factors give to the particles the physical properties of the sand.

In more details, the normal factor F_n is calculated as follows:

$$F_n = k_n(a_i + a_j - r)^{3/2} - \gamma_n m_e(v \cdot n)$$

where:

k_n	The elastic constant of the sand
γ_n	The constant that controls the amount of energy dissipation
m_e	The effective mass $m_i m_j/(m_i + m_j)$

The first part of this equation is the three dimensional Hertzian repulsion which is caused by the elastic deformation of the sand grain when it comes in contact with another sand grain. The second part is the friction term which depends on the relative velocity and is used for the dissipation of the energy in the simulation.

The shear factor F_s of the internal force is calculated by the following equation:

$$F_s = -\gamma_s m_e (v \cdot s) - \mathrm{sign}(\delta s)\min(k_s \delta s, \mu F_n)$$

where:

k_s	The restoring constant of the shear force
γ_s	The constant that controls the amount of energy dissipation
δ_s	The total shear displacement
μ	The static friction constant

The first term of this equation is similar to second part of the previous equation and defines the dissipation of energy during the contact. The second term is actually the implementation of the static friction, a physical property of the sand material. The general idea behind the static friction is the creation of a virtual spring in the direction of the shear force between two particles in contact. So, a restoring force is acted during the contact with a maximum value that is given by Coulomb's criterion. When the particles are no longer in contact, this virtual spring between them is removed. It is important also to note that the rotation of particles is not included in this simulation.

4. GROUND GRANULAR MODEL

4. 1 Representation model

In this thesis a hybrid approach for sand animation is proposed combining particles with a general model of deformable granular material. This model is represented by a heightfield which is defined by vertical columns. It is actually a continuous volume of sand on the ground that is divided into a rectilinear grid. Each column in the grid takes a defined height in order to form the necessary heightfield for the simulation. This heightfield replaces the invisible and static layer of a sand surface and, therefore, the necessary number of particles needed for the simulation is reduced by a significant margin. The resolution of the grid can be suitably selected according to the environment and the size of the objects that are going to interact with it.

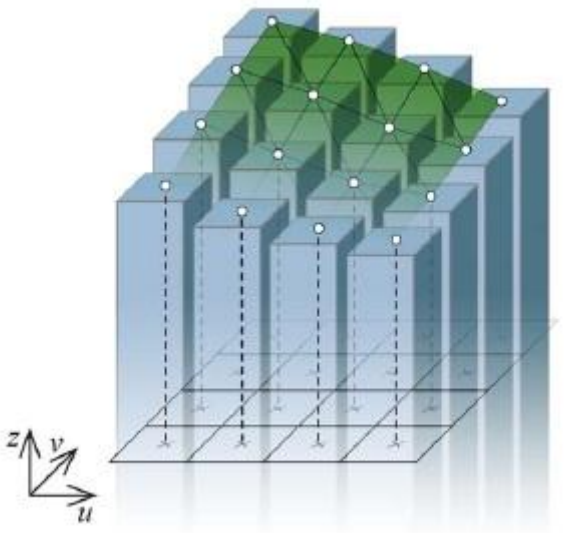


Figure 3. Representation of Heightfield (Holz et al,2009)

In this simulation, the heightfield has a predefined height for each column and it forms flat surfaces. But the initial conditions for the height of each grid point could also be created either procedurally or imported from a variety of sources. Unfortunately, there is not such an implementation in this simulation due to time constraints. But it is a challenge to be created in the future.

After the initialization of the heightfield, an algorithm for its interaction with objects in the environment needs to be implemented. This algorithm has three basic stages: collision detection, displacement, erosion. The collision detection between the heightfield and the objects in the environment will be discussed in the next chapter.

4.2 Displacement

The displacement of the sand surface is taken place when an object is in contact with the columns of the heightfield. Therefore, when a collision is detected, the columns that are in contact with the object are moving downwards until the object remains stable. In order to find the neighbouring columns that are not in contact with the object, an array is used to store a integer value. This array actually shows if a column is in contact with the object or not. If it is in contact, then it takes the value of 1, otherwise it takes the value of 0. This is useful when we try to find the surroundings columns for the displacement step of the algorithm.

This procedure is applied for every column of the heightfield that is affected by the surface of the object that is in contact with the heightfield. The material from these columns is either compressed or distributed to the surroundings columns that are not in contact with the object. This material is represented by the vertical displacement of the columns and is stored in a value (m) that is going to be used for the total distributed material in the neighbouring columns. For the compressed material, a ratio α is used and can be chosen from the user according to the type of the surface that is going to be appeared. Then the distributed material is calculated as follows:

$$\Delta h = \alpha * m$$

After these calculations, the height of each surrounding column that is not in contact with the object is increased in order to simulate the displaced material. The material is equally distributed in these columns and the next step is the procedure of erosion which is explained below.

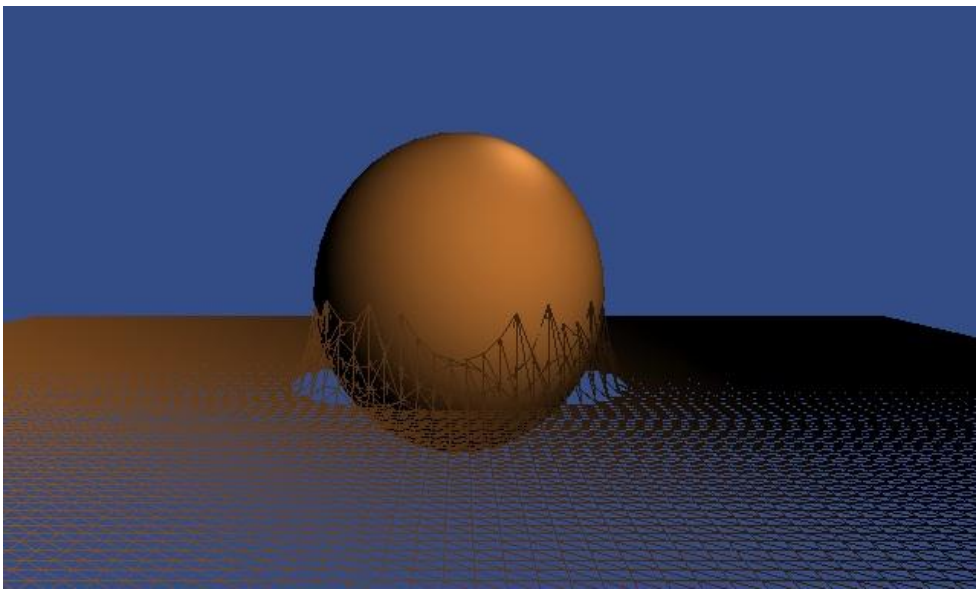


Figure 4. Displacement of surface

4.3 Erosion

After the material is displaced in the neighbouring columns, an erosion procedure needs to be applied for creating realistic deformation of the sand surface. After the displacement of the surface, the material is distributed only in the first ring of the surrounding columns that are not in contact with the object. Therefore, an unrealistic visual result needs to be corrected. During the erosion step of the algorithm, the steep slopes between the columns and their neighbours are examined and an amount of the material in each column is further moved to columns with a lower height.

The slope between two adjacent columns is actually represents the angle of repose of the sand. When the columns are moving upwards due to the transferred material of the compressed columns, sand piles are formed around the object on the surface. According to the physical properties of sand as a granular material, the height of a sand pile is dependent of the angle of repose. Therefore, each displacement in the columns of the heightfield needs to be further examined in order to produce realistic visual results close to physical accuracy. This is a reason why the slope between the adjacent columns has a significant role in the deformation of the sand surface. According to Sumner et al. (Sumner et al,1999), for a column ij and a neighbouring column kl, the slope, s, is calculated as:

$$s = \tan^{-1}(h_{ij} - h_{kl}) / d,$$

where h_{ij} and h_{kl} are the height of columns ij and kl respectively, and d is the distance between the two columns.

The next step after the calculation of each slope is the correction of any large values of slopes. This correction is implemented by defining a threshold (θ_{out}). If the slope between two columns is greater than this threshold, then the material is moved from the higher column down the slope to the lower column. This procedure is repeated until all slopes are below a second threshold (θ_{stop}).

The amount of material that is distributed between the columns with steep slopes is computed by the average difference between the neighbouring columns. The equation is:

$$\Delta h_a = \frac{\sum(h_{ij} - h_{kl})}{n},$$

where n are the neighbouring columns. The average difference in height is then multiplied by a fractional constant (s). The material after the above calculations is then equally distributed to the neighbouring columns with lower height. The erosion step of the algorithm gives the realistic visual results of a sand surface and by changing the factors, the appearance of the deformation on the surface is going to be different.

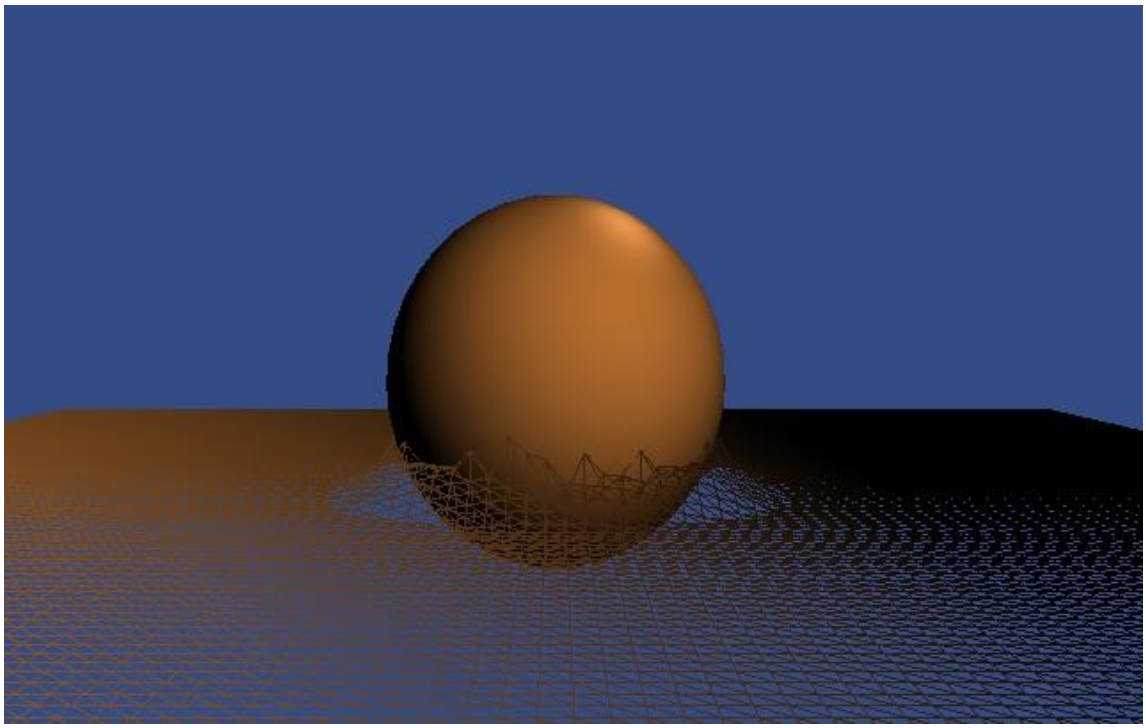


Figure 5. Erosion on the surface

5. COLLISIONS

In a physics-based simulation the management of different types of collision between the elements that take part in the simulation is a significant factor for the success of realism. Each collision has two main parts: the collision detection and the collision response. Both procedures increase the computational cost of the simulation significantly and, thus, they need to be handled carefully and effectively in order to succeed physical accuracy in the simulation. In the following chapters, the different types of collision in this simulation are going to be explained.

5.1 Particle-Particle Collision

For the representation of a sand grain, a spherical particle is used in order to simplify the collisions between each sand grain. By using spheres the collision detection between each pair of particles becomes very simple as we only need to compute the distance between the centres of each pair of particles. Consider two particles i,j with centres at positions p_i and p_j respectively. Given the radius r, as each particle in this simulation has the same radius, a collision occurs when:

$$dist <= minDist^2 \text{ where } dist = relPos^2 \text{ , } relPos = P_1 - P_2 \text{ and } minDist = r_1 + r_2$$

The collision response in each case is described by the contact model between two particles in the DEM algorithm above.

5.2 Particle-Object Collision

In this simulation, spherical objects in the environment have been used in order to lower the computational cost of the collisions between a large number of particles and the objects in the environment. The collision detection is the same as in particle-particle collision case but the collision response is a bit different. In this case, when a collision occurs, an additional force from the contact with the spherical object is added in the total sum of forces that affects the acceleration of each particle. This force is divided into two components, the normal and the tangential force.

Both components are calculated as described in the contact model with the static friction applied on the surface of the spherical object. But in this type of collisions, a different value for the elastic constant of the material (k_n) is defined depending on the surface of the spherical object.

5.3 Particle-Plane Collision

Another type of collision in this simulation is between the particles and a flat surface. Flat surfaces represent the walls of a bounding box of the upper face of the columns in the heightfield. In order to examine if a collision occurs between a spherical particle and a plane, the dot product (D) of the plane's normal P_n with the position vector of the centre of the particle (p) needs to be calculated as shown below:

$$D = P_n.dot(p)$$

Then it is checked if the value of D is greater than the radius of the particle in order to determine if there is collision or not. An additional check need to be considered, in order to define that the particle is inside the dimensions of the plane.

5.4 Heightfield-Object Collision

An important event of this simulation is happening when an object in the environment is falling on the sand surface that is represented from the heightfield described in the previous chapter. The collision detection between the object and the heightfield was a changeling task, as it is vital for the implementation of the ground granular model.

As far as the collision detection is concerned, a ray is cast from the vertex at the top of each column of the heightfield along the normal direction of the column. Because we use spherical objects in the environment, the collision can become a ray-sphere collision which is described in details below. First of all, the object is defined as a sphere with its centre position at a point $p(x, y, z)$ and a radius r. Then we consider as p_c the centre position of a sphere at a point (x_c, y_c, z_c). Given the equation of the sphere as:

$$x^2 + y^2 + z^2 = p \cdot p = r^2 \text{or } p \cdot p - r^2 = 0$$

we define the equation of a sphere centred at p_c as

$$(p - p_c) \cdot (p - p_c) - r^2 = 0$$

So, having a ray in the direction (d) of the normal of each column of the heightfield (d_x, d_y, d_z), parameterized by t, and given the starting point (p_0) of the ray as the position of the centre of each column, we get p in above equation as $p = t_d + p_0$.

So the sphere equation transformed into the following equation:

$$(td + p_0 - p_c).(td + p_0 - p_c) - r^2$$

Having defined all the above values, a hit is examined by solving the quadratic form of this equation. The discriminant of the quadratic equation is calculated as:

$$a * t^2 + b * t + c = 0$$

$$D = \sqrt{b^2 - 4 * a * c}$$

Now we check if the value of computed discriminant is greater than 0 or not. If D>0, it means that an object is detected in the direction of the ray and by solving the equation, the hit points can be found. By computing the distance between the first hit point and the height of the column, collision detection is occurred when this distance is equal to zero. In the opposite case (D<0), no collision is detected and, therefore, no further calculations needs to be done. The ray-sphere algorithm used in this simulation is as follows:

```
A = m_heightfield->m_Normal.dot(m_heightfield->m_Normal);
t=m_heightfield->m_heightPos-m_object->m_Position;
B= m_heightfield->m_Normal.dot(t)*2;
C=t.dot(t)-r*r;
discrim = B * B - 4*(A * C);
```

In order to restrict the calculations for the ray-sphere collision, the bounding box of the spherical object is projected into the surface and, as a result, only the columns that are inside the dimensions of the bounding box is considered in the collision detection algorithm. So the computational cost is reduced for this type of collision.

As far as the collision response in this case is concerned, an opposite force from the ground surface is applied to the object with the same magnitude as the gravity of the object. Due to the granular material of the surface, a dissipation of energy is happening and, therefore, the object is sinking in the surface until its vertical velocity becomes zero. This dissipation factor is applied to the velocity of the object and can be defined by the user according to the desired type of sand surface in the simulation.

6. SIMULATION

6.1 Spatial Hashing

An important part of any physical-based simulation is its optimization. In this sand simulation, the use of a large number of particles demands the computations to be as few as possible. One computationally expensive procedure in this simulation is the search for the neighbours of each particle. This procedure is performed in every time-step of the simulation and it actually affects the physical accuracy of the simulation, because it determines the particles that are going to interact with the internal forces. For this reason, a spatial decomposition has been implemented to reduce the time of this procedure.

This spatial decomposition has been succeeded using a spatial hashing algorithm developed by Teschner et al. (2003). The use of this algorithm for the optimization of the neighbouring search procedure has a great advantage. It actually decreases the complexity from $O(n2)$ to $O(nm)$ when the simulation needs to find the neighbours of each particle in every time-step.

The spatial hashing algorithm that is used in this simulation uses a hash function that converts the position vector of each particle to an integer hash value. This value is then stored in the suitable cell in the hash table. So, the search for neighbours becomes a procedure of finding the particles that are inside the same cell in the hash table. One important parameter in the algorithm is the appropriate selection of the cell size and it depends on the radius of the particles that is used in the simulation.

After having defined the cell size, a hash function is used in order to create the values that correspond to each position. Particles with same hash values are stored in the same cell. This hash function takes as input the discretised components of the 3D position vector (p) of the particle and it returns a value as follows:

$$hash(\hat{\mathrm{r}}) = (\hat{\mathrm{r}}_x p_1 \text{ xor } \hat{\mathrm{r}}_y p_2 \text{ xor } \hat{\mathrm{r}}_y p_3) \mod n_H$$

where: - r_x, r_y, r_z are the discretised values of p_x, p_y, p_z respectively.

These discretized values are the result of a function that computes these values as follows:

$$\mathbf{r}(\hat{\mathbf{r}}) = (\lfloor \mathbf{r}_x/l \rfloor, \lfloor \mathbf{r}_y/l \rfloor, \lfloor \mathbf{r}_z/l \rfloor)^T$$

- p_1, p_2, p_3 are three large prime numbers that are proposed from Teschner et al. (2003) as:

$$p1 = 73856093$$

$$p2 = 19349663$$

$$p3 = 83492791$$

- n_H is the size of the hash table and it is a predefined prime number, in order to generate the right values for each particle's position.

The next step is the use of the hash function for all particles in the simulation. By generating the hash value for each particle, the insertion of each particle in the hash table is taken place. In this simulation, the hash table has been implemented as a *std::multimap* that takes a pair of an integer (hash value) and an instance of the particle (*particle). Finally, the search for the neighbours of each particle is restricted in a search of particles with the same hash value from the hash map that is created. This spatial hashing algorithm is repeated in every time-step and the hash map is refreshed as the position of the particles has changed as the simulation is running.

6.2 Integration

The motion of each particle in the simulation is dependent on the acceleration vector in each time-step. The acceleration can be calculated using Newton's second law of motion:

$$a = F/m$$

where *F* is the total force that is applied on the particle and *m* is the mass of the particle.

The integration of the acceleration is vital for the motion of the particles as it is used in order to calculate the velocity and the position of the particle in each time-step. In this simulation two

integration methods has been tested: semi-implicit Euler and Verlet integration.

6.2.1 Semi-implicit Euler

Euler Integration is one of the most basic integration methods to implement. It is based on Newton's second law of motion and uses the notion of derivatives to find the velocity and the position. While the energy is conserved in each time-step, this method appears several numerical instabilities in this simulation and is not always preferred. The next position of each particle is implemented with the following algorithm:

$$m_{oldPosition} = m_{Position};$$

$$m\_Velocity \mathrel{+}= m\_Acceleration;$$

$$m\_Position \mathrel{+}= m\_Velocity;$$

6.2.2 Verlet Integration

Verlet Integration is a fast method for numerically integrating the equations of motion for the movement of each particle in the simulation. It is quite stable as it bypasses the calculation of velocity and calculates the next position directly from the acceleration. It is also useful as it stores the old and the new position of the particle, as this information is necessary for the calculation of the shear force when the particle is in contact with another particle. The algorithm that was used for the implementation of verlet integration in this sand simulation is shown below:

$$next\_position = current\_position + (current\_position - old\_position) * (1 - dissipation) + acceleration$$

6.3 Hybrid Approach

The two main models that are implemented in this simulation have been described in the previous chapters. The DEM algorithm is responsible for the simulation of the particles and their interactions with the environment and themselves. The ground granular model is responsible for the simulation of a sand surface. The changeling part of this thesis project was to combine these two models into one. This hybrid approach is going to speed up the sand simulation without losing in physical accuracy. The DEM method has a significant computational cost and when the number of particles becomes pretty large, the simulation runs in a low frame rate. In contrast, the ground material model with the use of heightfield is a suitable model for real-time deformations of a surface. Therefore, the combination of these two parts would create an interesting result.

By taking advantage of the fact that the sand is an opaque material, the static and invisible particles in a sand pile or a sand surface with a heightfield. As a result, the computations in every time-step of the simulation are decreased and the simulation runs in an acceptable rate of frames per second. The algorithm of the replacement of these particles is the follow one:

In every update of the simulation, the distance between the current position and the previous position for each particle is being checked. If this distance becomes very small and approaches to zero, then the particle is defined as inactive. Being inactive means that the particle is stable in the simulation and there is no need for further calculations of the total force that is applied to it until another new particle interact with it. So, the inactive particles that are not on the outside layer of the surface of the sand pile are candidates for replacement.

The algorithm for the creation of the heightfield and the replacement of the inactive particle s checks all the candidates and it creates vertical columns in the position of the replaced particles. The height of these columns defines the volume of the replaced particles. So the total volume of the sand is preserved with fewer particles in the current time-step. The heightfield that is formed in this case belongs to the ground material model that is described in chapter 3. Therefore, all the physical properties of sand are still there with a lower computational cost than before, as a smaller number of particles is now active in the simulation.

6.4 Visualisation

For the visualisation of this simulation, simple OpenGL representation models have been used. Due to time constraints, the visualisation of this project was not a primary task as the most amount of time was spent in the implementation of the functionality of the sand simulation. Nevertheless, the sand particles are represented either as spheres in one scene, where the number of particles is not too big, or as points, where a large amount of particles are needed for the creation of the sand surface. The heightfield is represented as a 2D grid and the vertices of the height of each column are connected in order to form a triangulated surface. A simple shader is also used that gives a defined colour to particles via the ngl library.

Three scenes have been created for the visualisation of this simulation. In the first scene, particles of sand are falling from a predefined height in a constant rate and they form a sand pile when they touch the bottom surface of the box. In the second scene, the sand particles are falling from the same height but now there is a spherical object in the scene that interacts with them. In the third scene, the particles form a sand surface and an object is falling causing deformations on the sand surface. All these three scenarios are suitably chosen in order to show the functionality of this simulation and to test the physical accuracy of sand as a granular material.

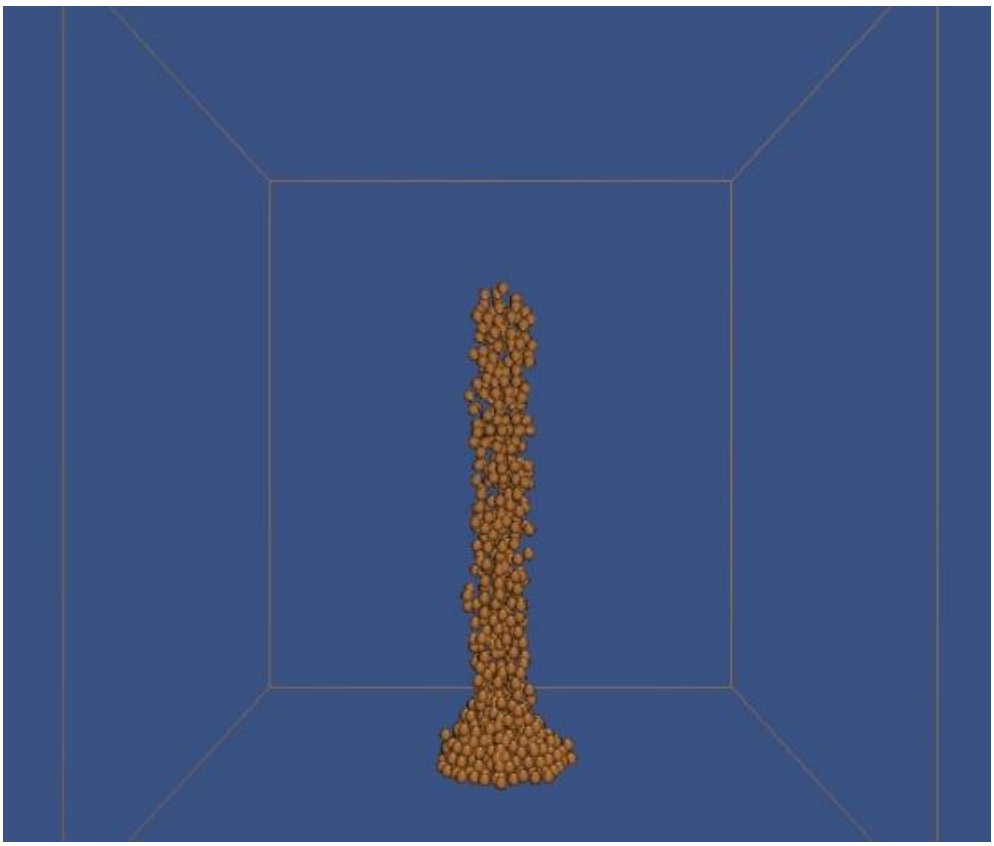


Figure 6. Scene 1

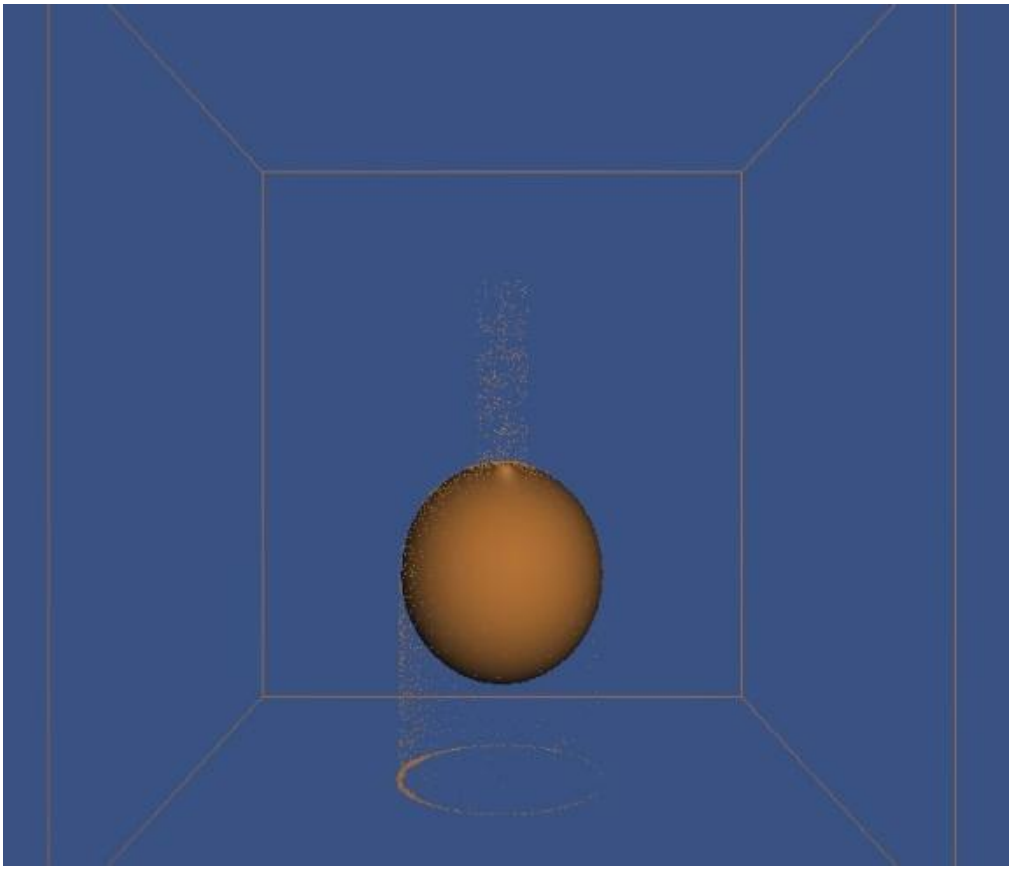


Figure 7. Scene 2

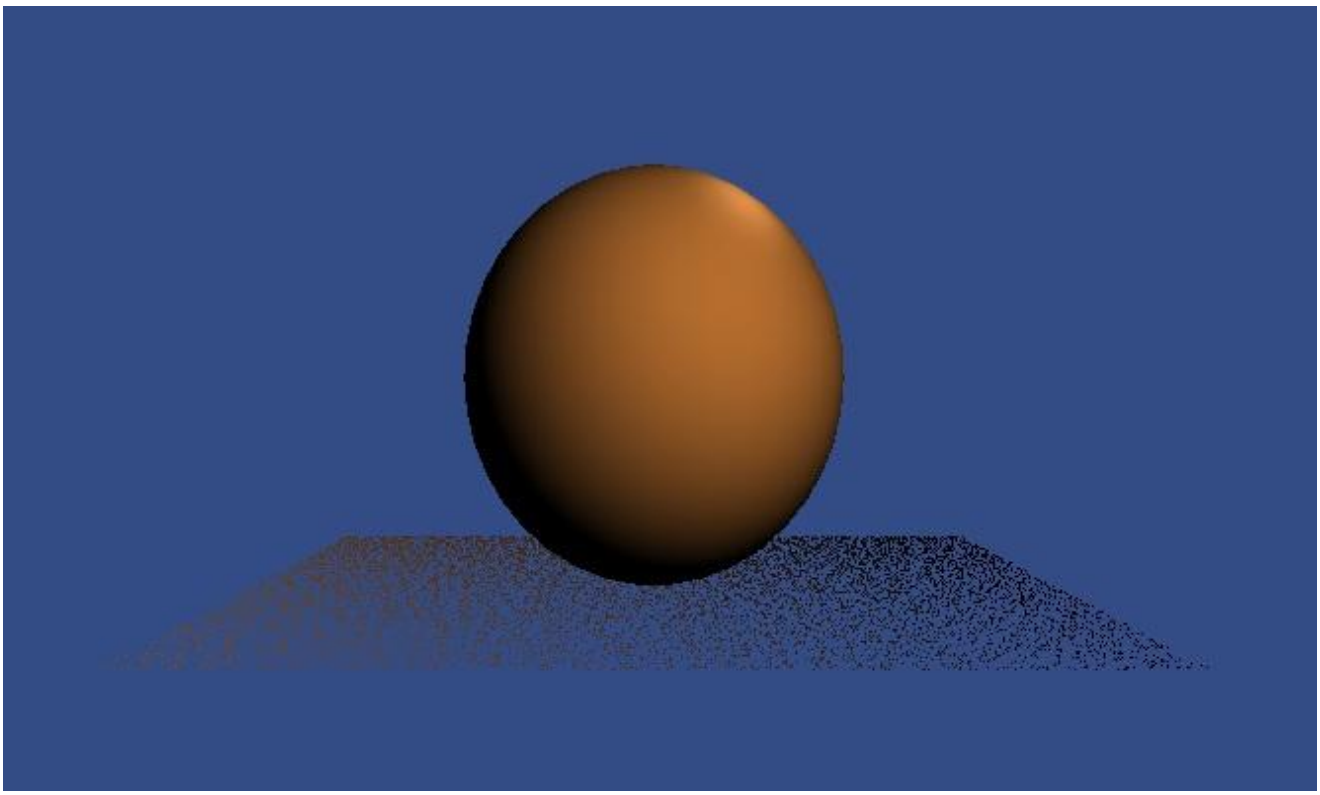


Figure 8. Scene 3

7. CONCLUSION

7.1 Performance

The implementation of sand simulation was a changeling task from the very beginning of the project. Sand has unique physical properties that are different from other materials and, as it has mentioned before, it cannot be simulated in real-time. Using a hybrid approach by combining a particle system with a mesh represented by a heightfield was an effective way of simulating sand particles in pretty good frame rates.

In this simulation, the Discrete Element Method is used for the implementation of the necessary forces that are applied to the particles. The DEM method is very similar to Molecular Dynamics methods that perform the calculations for the motion of each element in each particle in the system. It has been previously found that the DEM method is an accurate approach to the physical properties of the granular materials, such as the sand. Therefore the simulation in this project appears a nice visual result in terms of physical accuracy.

By using the ground model, the simulation gains in speed without losing physical accuracy. This was tested by comparing the creation of a sand pile when one thousand particles are poured from a defined height. By using the ground model, the particles that are invisible and stable are replaced by the ground model that was described before. So the time of the simulation is improved by almost 30%. The basic advantage of this hybrid approach is the fact that a significant amount of sand particles with computationally expensive contact forces do not need to be considered for the appearance of the simulation as they are replaced by the heightfield. Therefore, the simulation does not suffer from the rather small time steps that the DEM usually requires. In addition, the computational cost of the ground granular model is trivial in comparison with the DEM model. However, if more particles are used, the improvement in terms of run time is decreasing. Nevertheless, this speed up in the simulation is significant and can be improved more in the future.

7.2 Limitations

During the implementation of this simulation, many limitations have been appeared. First of all, the issue of handling a ton of particles was the main problem of this project. The DEM method is a computationally expensive and in most cases is not used in real-time applications. Despite the use of some optimizations techniques, which were explained before, the simulation cannot run in an acceptable frame rate if the number of particles is increased significantly. This issue could be overcome in an extent by the use of parallelisation of the algorithms. Maknickas and Kaceniauskas (2006) proposed a parallelisation of the DEM method with pretty good results. The speed up in the simulation running time in this project could be succeeded by the use of parallel programming in CPU or even better in GPU.

Another limitation of the use of this simulation is the missing of an appropriate visualisation. In the initial proposal of this thesis, a implementation for the Maya software was proposed for the visualisation of the simulation. However, due to difficulties of the hybrid approach and the combination of two different systems, the implementation via Maya API C++ was rejected. Furthermore, only spherical objects can be used for the interaction with the sand in this simulation. The use of polygonal objects was not a primary task for this project. But with the appropriate algorithm, any object can be represented by spherical particles and can be used for testing with the sand in this project.

7.3 Future Work

The sand simulation in this project using the proposed hybrid approach offers a nice visual result in terms of physical accuracy with a respectable run time. Of course, many improvements can be taken place for improving this technique in order to be applied in more complex scenes. A significant improvement for this technique will be a parallel implementation in the GPU of the described algorithms using OpenCL or CUDA. By taking advantage of the speed up that the graphics card's processors offer, the sand simulation could be run in close to real-time speed for a

large number of particles.

Furthermore, the nature of the project has a great potential for more work. Not only would a parallel implementation boost up the speed of the simulation, but also the finding of a quicker way to identify the neighbours of each would also be quite effective in terms of computational cost. The spatial hashing algorithm that is used in this simulation is a good approach, but when the number of particles becomes extremely big, this algorithm has its limitations. Finally, a way of handling more complex objects in the environment would be a good consideration as future work. For now, the simulation can handle only collision and interactions with spherical objects. By adding algorithms of collision detection and response with complex polygonal objects would add interesting results to the application.

REFERENCES

Alduan, I., Tena, A., and Otaduy, M. A. 2009. Simulation of high-resolution granular media. In *Proc. of Congreso Espanol de Informatica Grafica*.

Ammann C., Bloom D., Cohen J., Courte J., Flores L., Hasegawa S., Kalaitzidis N., Tornberg T., Treweek L., Winter B., Yang C.: The birth of sandman. *ACM SIGGRAPH 2007 sketches* (2007).

Aradian A., Rapahel E., and De Gennes, P. Surface flows of granular materials: A short introduction to some recent models. *Comptes Rendus Physique 3* (2002), 187–196. 10 pages, 3 figures, published in a special issue of C. R. Physique (Paris) on granular matter.

Benes, B. and Roa., T. Simulating desert scenery. *Winter School of Computer Graphics SHORT communication Papers Proceedings*, pages 17–22, 2004.

Benes B., Dorjgotov E., Arns L., Bertoline G.: Granular material interactive manipulation: Touching sand with haptic feedback. In *Proceedings of the 14-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006* (2006), pp. 295–304.

Benes, B. and Forsbach., R., Layered Data Structure for Visual Simulation of Terrain Erosion. In Tonisao Kunii, editor, *IEEE Proceedings of the Spring Conference on Computer Graphics*, pages 80–86. IEEE Computer Society, 2001.

Duran, J., 1999. Sands, Powders and Grains: An Introduction to the Physics of Granular Materials. Springer.

Hudak, M. and Durikovic, R., 2011. Terrain Models for Mass Movement Erosion. In *Theory and Practice of Computer Graphics*, pages 9-16, 2011.

Holz, D., Beer, T. and Kuhlen T., 2009. Soil Deformation Models for Real-Time Simulation: A Hybrid Approach. Workshop on Virtual Reality Interaction and Physical Simulation.

Macey, J. (2010). Ngl graphics library. Available from: http://nccastaff.bournemouth.ac.uk/jmacey/GraphicsLib/ [Accessed 19 Aug 2012].

Lenaerts, T., and Dutre, P. 2009. Mixing fluids and granular materials. *Computer Graphics Forum 28*, 213–218(6).

Narain, R., Golas, A. and Lin, M., Free-Flowing Granular Materials with Two-Way Solid Coupling. In *Proceedings of the 2010 ACM SIGGRAPH Asia*, Volume 29, Issue 6, December 2010.

Nathan B., Yizhou Y., and Mucha., P. Particle-based simulation of granular materials. In *Proceedings of the 2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 77–86, July 2005.

Onoue, K. and Nishita, T. Virtual sandbox. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 252. IEEE Computer Society, 2003.

Parent, R., 2008. Computer Animation: Algorithms and Techniques. 2nd Edition. Massachusetts: Morgan Kaufmann.

Pla-Castells, M., Garcia-Fernandez, I., and Martinez, R., 2006. Interactive terrain simulation and force distribution models in sand piles. In *Cellular Automata*. 392–401.

Pla-Castells, M., and Garcia-Fernandez, I., Physically-based interactive sand simulation. In *Eurographics 2008 - Short Papers* (2008), pp. 21–24.

Sumner, R., O'Brien, J., and Hodgins, J., 1999. Animating sand, mud, and snow. *Computer Graphics Forum 18*, 1, 17–26.

Teschner, M., Heidelberger, B., M□uller, M., Pomeranets, D., and Gross, M. (2003). Optimized spatial hashing for collision detection of deformable objects. In Proceedings of Vision, Modeling, Visualization VMV03, pages 47{54. Available from: http://www.beosil.com/download/CollisionDetectionHashing_VMV03.pdf [Accessed 19 Aug 2011].

Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Trans. Graph.*, 24:965–972, 2005.

Zhu, B., and Yang, X. 2010. Animating sand as a surface flow. In *Eurographics 2010, Short Papers.*