Wet sand simulation in Maya

MSc Computer animation and visual effect

Quan Xiang

Introduction……3

Previous……

…4

Technique

background……5

Granular particles simulation……5

Fluid

simulation……7

interactions and wetness value between fluid and granular particles……

…9

Propagation of the wetness attribute……

13

Maya

API/Mel……14

Project implementation and conclusion……16

Objective……

16

The approach……

16

Implementation process……

17

Conclusion……

18

Reference……

…19

Appendices……

20

Introduction

People who love to observe the nature may notice that when sand or soil become wet, they become sticky at the same time; children even can build up sand castles by wet sand. That because when sand becomes wet, the small space between each sand particle is filled with fluid and stick sand particles together by “liquid bridges”, but when the sand particles are overwet, the liquid bridges will lose the their power, the sand sculptures will collapse because of losing their internal forces and collapse.

As an popular 3D software, Maya provides a group of effective fields which work on both particles and normal polygon or NURBS objects, they are so powerful that can achieve most of the main nature phenomena, however, a custom field node will efficient Maya's function.Although it seems the latest Maya2009 has already add this function , Maya2008 which version I made use of in the project only provide "make collide " function that handling the collision events between particles and geometric objects, the collision between two groups of particles cannot be dealt with. However in my project, the collision detection for two groups of particles is necessary, so the first part of the Mel file is used to detect it. The detection result determines the destiny of particles, in this project;, particles will either increasing its wetness value or becoming dead. when those alive particle's wetness attribute reach a level, their would like to pass their wetness to their adjacent particles, this is called "wetness propergation", which is an important feature of the granular particles.

Previous work

There are many articles and reports, discus about sand simulation and fluid simulation.

"Physically-based interactive sand simulation" from M. Pla-castells (2008) discussed about the basic principles and scheme of physically-based simulation. In this article, sand pile is simulate like a whole object, there

is no force between sand pile itself, this method is mainly used when only considering outside forces affect the sand system.

In “Particle-based simulation of granular” material from Nathan Bell (2005), each granular particle is treated as independent rigid body, in this article, internal forces affect exist among sand particles except forces from outside environment.

“Particle-based fluid simulation for interactive applications” from Muller M (2003) introduced and post large amount of valuable fluid simulation principles and equations, another article, “Real-time animation of sand-water interaction” also cite Muller’s research result.

“Real-time animation of sand-water interaction” from Witawat Rungjiratananon (2008) is the main reference for the project, all the algorithm and equations in the project are based on the content of this article.

Technical Background

1. granular particles simulation

Granular materials, such as grains, soil and sand are widely used in animations and movies recent years, to simulate realistic phenomena

effects. Compare with other particles, for instance, smoke and fireworks, granular particles have their own properties, such as mass and surface tension. In some extension, each granular particle is like a tiny rigid body. There are two main techniques are widely used to simulate granular material: particle-based simulation and physically-based simulation.

1.1 Physically-based simulation

In this method, classically discrete theoretical models are adopted to simulate granular material as a part of environment like terrains or sand piles (M, Castells, 2008). Physically-based approach is based on the behaviors of the environment, they usually are used when the grain particles have interaction with their around environment. In this approach, all granular particles are treated as one object.

1.2 Particle-based simulation

This approach represents granular particles as a large collection of discrete elements. The advantage of using discrete elements is that pre-particle is able to inter interactions with other particles and environment around it, and there is advection effect among particles. Compare with physically-based simulation, this approach conveniently generates some highly dynamic phenomena without sacrificing physical accuracy.

Different with physically-based simulation, particle-based employs Molecular Dynamics (MD) method. This method permits small overlap between particles, to simulate the elastic deformation of material elements and to determine appropriate internal-contact forces. So unlike physically-based method which has to consider collisions and horizontal displacement, and reaction forces that totally from outside and worked on the how simulation system, at particle-based simulation, per particle has to consider collisions and frictions between themselves and contact force from outside larger objects.

the very first step is detecting collision fluid particles and granular particles, after that is the really surface colour changing process.

1.3 implementation of physically-based simulation and particle-based simulation

Physically-based: as the cellular automata formulation on a ordinary grid which is managed as granular object is adopted to fulfill the simulation model, detecting and keeping the grid's active grid cells list is the key of this simulation, as it only checked and updated active grid and react them, all the other negative grids are ignored. This scheme can save a large amount of computation.

Particle-based: it models a grain as multiple round particles that are constrained to move together as a single rigid body, define the relevant physical quantities and then examine a common force schemes. Simulate the interation between granular particles are related to contact forces.

2. Fluid simulation

Lagrangian's fluid simulation technique approximates a continuous fluid using a set of particles and models their behavior (real-time animation). It does not have numerical dispersion, this feature makes the technique useful for simulating when large changes happens in the fluid interface.

There are two significant ways for fluid simulations, smoothed particle hydrodynamics (SPH) and the Moving Particle semi-implicit (MPS). The former solves the particle advection equation explicitly; it is suitable for computer graphics due to its computational simplicity.

2.1 SPH (smoothed particle hydrodynamics)

SPH is a particle-based simulation technique, was originally developed for astronomy, in this method, particles are thought as a discrete approximation of a continuum, to affect a field quantity A(x), SPH method has:

$$A(\mathbf{x}) = \sum_i m_i \frac{A_i}{\rho_i} W(\mathbf{x} - \mathbf{x}_i, h),$$

Where mi : the mass of particle i;

ρ_i : the density of particle i;

xi: the position of particle I;

W(x-xi, h): a smoothing kernel function;

H: core radius h, or bandwith.

2.2 DEM(Discrete Element Method)

DEM is also particle-based. For particle i and particle j, when they have collision, their normal forces and tangential forces is acting.

$$\begin{aligned}
\mathbf{F}_i^{normal} &= \mathbf{F}_i^{spring} + \mathbf{F}_i^{damper}, \\
\mathbf{F}_i^{spring} &= k_s(d - \|\mathbf{x}_i - \mathbf{x}_j\|) \frac{\mathbf{v}_{ij}^{normal}}{\|\mathbf{v}_{ij}^{normal}\|}, \\
\mathbf{F}_i^{damper} &= k_d \mathbf{v}_{ij}^{normal}, \\
\mathbf{F}_i^{tangential} &= k_t \frac{\mathbf{v}_i^{tangential}}{\|\mathbf{v}_i^{tangential}\|},
\end{aligned}$$

Where ks : the spring coefficient;

d: radius[i] + radius[j];

kd: the damper coefficient;

kt: the coefficient coefficient;

vnormal: particles' normal velocities

vtangential: particles' tangential velocities

3. interactions and wetness value between fluid and granular particles

3.1 liquid bridge

when a fluid particle encountered with a granular particles, a physical mechanism will occur: fluid particles will absorb into the tiny overlaps between granular particles, generate many “liquid bridges”. These liquid bridges are from the surface tension of the fluid particle, they assign granular particles internal attractive forces, let wet granular particles stick together. These liquid bridges provide forces and cohere those granular particles. With the help of liquid bridges, in this stage, wet sand can be dug out to create sand art sculptures.

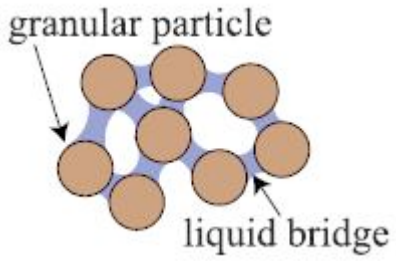

Figure1: liquid bridge and granular particles

3.2 wetness changing

To understand and make use of the features of liquid bridge, we define a wetness value for both fluid and granular material particles. Of course, at the very beginning, fluid particles have a max wetness value, while granular particles have a zero wetness value, in this stage, granular particles are “dry”. When collision happens, granular particles gain wetness value from the encountered fluid value. The process is shown below:

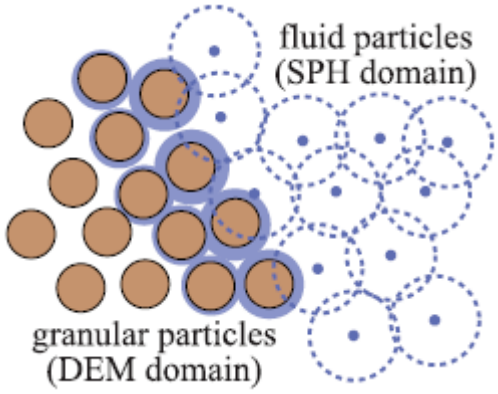

Figure2: granular particles gain wetness.

However, the wetness value has a threshold, when a granular particles wetness value is larger than zero but less than the wetness threshold, it obtain attractive forces with others granular particles who are linked with it by the same liquid bridges, and cohesion strength increased as well; in this situation, the wetness value is greater, the forces is stronger, in this stage, granular particles keeps on receive wetness value and because of wetness was deprived, fluid particles disappear, in this stage, granular particles are "wet". However, when the wetness value is over the threshold, the excessive part is distributed equally to the wetness owner's neighboring particles who have wetness values less than the threshold.

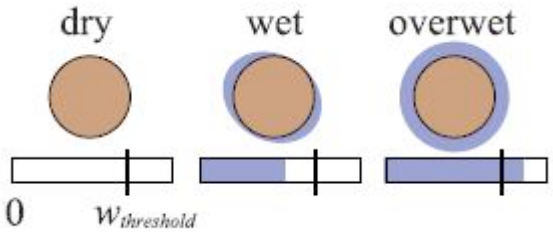

Figure3: different stages of granular particles wetness value

In the interaction simulation model, we define a wi that representing the real wetness value granular particle i, wi should be: 0 <= wi<= wmax,

then the pseudo code should be:

```
if (wi==0);
        dry particle;
else if (wi>0 && wi<=wthreshold )
        wet particle;
else // ( wi > wthreshold    && wi <= wmax)
        overwet particle;
```

the wetness value is used to generate and compute the attractive forces among granular particles, for representing the affection from the aggregation of wetness value of granular particles, the radius of a granular particles decressed.

$$r_i = R - k_r * w_i$$

where r_i is the current radius of particle i;

R is the original radius of particle i, which used in DEM calculation;

K_r is a coefficient.

When a fluid particle i and a granular particle j has collision, a method based on SPH formulation is called to compute the inter-particle forces

3.3 Interactions between fluid particles and granular particles calculation

When collision happens between a fluid particle i and a granular particle j, the inter-particle force is generated, which calculation is base on SPH

method, the equations are:

$$\mathbf{F}_{ij}^{pressure} = -\frac{V_i V_g}{4}(p_i + p_g)\nabla W(\mathbf{x}_i - \mathbf{x}_j, h),$$
$$\mathbf{F}_{ij}^{viscosity} = \mu\frac{V_i V_g}{2}\nabla^2 W(\mathbf{x}_i - \mathbf{x}_j, h),$$

Where Vi is the volume of fluid part i;

Vg is the Volume of the granular particle;

Pg is the constant pressure;

For granular particle j, negative pressure force and negative viscosity are added as external forces. (real animation).

After the calculation of above forces, the fluid particle will began to consider its absorption. if the granular particle is dry or only wet, the fluid particle is absorbed, its wetness will be gained by its neighboring granular particles.

3.4 interactions among granular particles calculation.

In the calculation, the equations that modified from DEM method related to the amount of wetness are used:

$$\mathbf{F}_i^{damper} = k_d(1 + w_i + w_j)\mathbf{v}_{ij}^{normal},$$
$$\mathbf{F}_i^{tangential} = k_t(1 + w_i + w_j)\frac{\mathbf{v}_i^{tangential}}{\left\|\mathbf{v}_i^{tangential}\right\|}.$$

The liquid bridge force is computed in this stage as well:

$$\mathbf{F}_i^{bridge} = k_{bridge}\max\left\{0, w_f - \frac{w_i + w_j}{2}\right\}(\mathbf{v}_j - \mathbf{v}_i),$$

Where kbridge is a coefficient of liquid bridge effect;

wf is the threshold value;

Unlike the damper force and the tangential force, the bridge force does not always exist, actually, it only exist when (vj –vi) (xj - xi) > 0

4. Propagation of the wetness attribute

After computing all above forces, the overwet granular particles should be dealt with. As mentioned in the previous, overwet granular particles will devote a part of their wetness value averagely to their neighbouring particles whose wetness values are not over the threshold value. The equation is following:

$$w_j^{t+1} = w_j^t + k_p \frac{\Delta w_i^t}{N_i} \Delta t,$$

Where Δt : timestep;

kp: coefficient used to control the propagation speed;

Δw_i^t :at time t, the total wetness value needs to devote to neighbor.

Ni: the number of neighbor.

5 Maya API / Mel

Maya is undoubtly a very powerful tool for 3D animations and visual effect in movies. It is a node based software, and has high programmability, it can be improved by either C++ with maya API or Maya Embedded language (MEL). Users can create their own user interface by MEL and using MEL to control all functions on the interface

by typing command in the scripting editor; they also can writing a node by C++ to generate plug-ins so that complete an task will become easier.

5.1 C++

Users can develop native plug-ins through C++ and Maya API, the task that maya API can do including:

a.) Commands: custom commands can be created to simplify a boring operation process by combine them together and typing one word or several short words in the scripting editor

b.) Dependency Graph Nodes: since Maya is a node basic package, creating new DG node means more functions or effects users can achieve in Maya.

c.) Tools/Contexts: in order to complete some hard operations, creating a custom tool will make work easier; and creating custom context allow users implement specific task , like specific modeling.

d.) File Translators: can support the compatibility between different package and file formats.

e.) Deformers: by developing custom deformer, users can deform object to specific shapes

f.) Shaders: using API and C++, users are able to improve their work's performance.

g.) Manipulators: it provide a series of visual controls.

h.) Locators: it helps user contol their object

i.) Fields: a field is typically applied to particles, append different types of forces to particle. By setting particles moving rules, user can achieve many visual effects based or generate by particles.

j.) Emitters: also used to control particles, users can changing emitters' shape, adding or deleting some attributes,

k.) Shapes: hold the actual geometry

l.) Solvers.

5.2 MEL

MEL is a strong scripting language in Maya, even the entire Maya GUI is written and controlled by it. All functions or effects users achieved can also be achieve by MEL programming. However, MEL does not allows users to creat custom commands.

5.3 simple comparison

Maya's C++ hierarchy provides access to assign Maya much more powerful functionality with the support from MayaAPI, but in many cases, MEL can provide strength functionality.

Only through the C++ API can user create their own nodes that allows users to creating all other types of custom tools. However, only MEL can create UI in Maya.

Project implementation and conclution

1. Objective:

The initial idea of the project is developing a plug-in that can cooperate with other Maya functions to generate sand-water interactions. When a granular material touches fluid, the first impression to people is that its colour becomes darker, since those grain to grain forces can be simulate by controlling current existing field, the colour automatically changing to let viewer know what happen immediately is an important part of the project's objective.

2. The approach

The final achieve, some mel files of the project should cooperate with maya's dynamic system. By creating two particles and adding a new per particle attributes in one of particles, the mel will effect the two particles and generate wet sand effect.

3. implementation process

from the previous technique research, particle-based simulation method undoubtly should be chosen, because the surface colour is per particle attribute, when a particle becomes wet, it doesn't meant that his neighbour must change their status..

after determine the basic approach method chosen, we will notice that wetness value cannot be ignored for the wet surface simulation,

so it is undoubtly that the particles which is the simulation of sand must add a new array attribute call wetnessPP to control each particles wetness value.

The basic condition that sand becomes wet is the sand particle has collision with fluid particles. Since maya does not have any existing function that can detect two particle groups' encounter, this detecting process must be completed by either C++ or MEL. As this is a simple task, MEL is powerful enough to deal with it. The basic principle of detecting collision is calculate distances among particles, if the distance is shorter than a constant value, then they collided.

The next step is change the texture or colour of the wet sand particles. Unfortunately, maya does not apply function that allows per particle has its own texture. The second option colour attribute has to be chosen to simulate the wet surface of a sand. Activate rgbPP attribute, check the value of wetnessPP, when it is in a different area, the rgbPP has different colour.

Except achieving the surface colour changing, to simulate the real process that sand becomes wet, wetness value must do some computation to achieve the wetness propagation effect. To

complete this task ,the neighbourhood of every particle must be checked, and using the wetness propagation equation mentioned in the technique background. When this function complete, the sand simulation animation will have wet sand propagation effect.

4. conclusion

although the simulation looks achieved the surface simulating wet granular material, the colour change is not smooth. We define constant colours at different stage, but in reality, it is smoothly, gradually changed, so in next step, the project can be improved to generate more realisitic result.

Reference:

[1] *M. Pla-Castells1. (2008)* Physically-Based Interactive Sand Simulation, *,The Eurographics Association.*

[2] *Nathan Bell, (2005)* Particle-Based Simulation of Granular Materials, *, ACM SIGGRAPH/ Eurographics Symposium on Computer Animation,.*

[3] *Yongning Zhu, (2005)* Animating Sand as a Fluid, *ACM SIGGRAPH,.*

[4] *Christoph Ammann, (2007)* The Birth of Sandman, *ACM SIGGRAPH.*

[5] *Witawat Rungjiratananon, (2008)* Real-time Animation of Sand-Water Interaction, *Pacific Graphics.*

[6]Parent,R., (2008). Computer animation: algorithms & techniques.2nd ed. San Francisco: Morgan Kaufmann

[7]David A.D.Gould, (2003), cComplete Maya Programming. Morgen

Kaufmann publishers, London,

[8]MARK R. W,(2003) Chris K, Scripting for maya animatiors Morgen Kaufmann publishers, London,

Appendices

A: MEL files:

wetsand_with_prop.mel: this mel file can achieve wetness propagation effect

```
//calculate distance
proc float distance(vector $fluid, vector $sand)
//detect collision
proc int hasCollision(vector $fluid, vector $sand, float $radFluid, float $radSand)
//change colour and lifespan when collision is detected_____pass
proc collisionDeform(float $fluid[], float $sand[], float $wetnessPP[], float $radFluid,
float $radSand, float $lifespan, float $dry,float $midwet, float $wetThres,float
$kp,float $offset)

//obtain the sand1's neighbour index array
proc int[] neighbour(vector $sand1, float $sand[], float $radSand)
```

wetsand_without_prop.mel : this mel file only has wetsand effect, no propagation

```
//calculate distance
proc float distance(vector $fluid, vector $sand)

//detect collision
proc int hasCollision(vector $fluid, vector $sand, float $radFluid, float $radSand)


//change colour and lifespan when collision is detected
proc collisionDeform(float $fluid[], float $sand[], float $radFluid, float $radSand,
float $lifespan, float $cR, float $cG,float $cB)
```

B: expressions used to test mel file

test_with_propagation.mel:

```
// obtain pre-particle position
float $fluid[] = `getParticleAttr -at position -array true waterParticleShape`;
float $sand[] = `getParticleAttr -at position -array true sandpileShape`;
float $sandWetness[] = `getParticleAttr -at wetnessPP -array true sandpileShape`;
//collisionDeform(float $fluid[], float $sand[], float $wetnessPP[],
//              float $radFluid, float $radSand, float $lifespan,
//              float $dry,float $midwet, float $wetThres,float $kp,float $offset)


collisionDeform($fluid, $sand, $sandWetness,
            0.3, 0.2, 0.2,0.0,
            0.3, 0.6, 0.8,0.5);
```

test_without_propagation.mel:

```
// obtain pre-particle position
float $fluid[] = `getParticleAttr -at position -array true waterParticleShape`;
float $sand[] = `getParticleAttr -at position -array true sandpileShape`;
float $sandWetness[] = `getParticleAttr -at wetnessPP -array true sandpileShape`;
//collisionDeform(float $fluid[], float $sand[],
//                  float $radFluid, float $radSand,
//          float $lifespan, float $cR, float $cG,float $cB)
collisionDeform($fluid, $sand,
            0.1, 0.1,
            0.0,0.39,0.287,0.185);
```