Vis Comput
DOI 10.1007/s00371-012-0765-1

ORIGINAL ARTICLE

Modeling and animation of fracture of heterogeneous materials based on CUDA

Jiangfan Ning · Huaxun Xu · Bo Wu · Liang Zeng · Sikun Li · Yueshan Xiong

Abstract Existing techniques for animation of object fracture are based on an assumption that the object materials are homogeneous while most real world materials are heterogeneous. In this paper, we propose to use movable cellular automata (MCA) to simulate fracture phenomena on heterogeneous objects. The method is based on the discrete representation and inherits the advantages from both classical cellular automaton and discrete element methods. In our approach, the object is represented as discrete spherical particles, named movable cellular automata. MCA is used to simulate the material and physical properties so as to determine when and where the fracture occurs. To achieve real-time performance, we accelerate the complex computation of automata's physical properties in MCA simulation using CUDA on a GPU. The simulation results are directly sent to vertex buffer object (VBO) for rendering to avoid the costly communication between CPU and GPU. The experimental results show the effectiveness of our method.

J. Ning (✉) · H. Xu · B. Wu · L. Zeng · S. Li · Y. Xiong
School of Computer Science, National University of Defense Technology, Changsha, China
e-mail: jiangfanning@gmail.com

H. Xu
e-mail: xxhhxx@163.com

B. Wu
e-mail: wubogfkd@yahoo.com.cn

L. Zeng
e-mail: liangzeng@263.net.cn

S. Li
e-mail: lisikun@263.net.cn

Y. Xiong
e-mail: ysxiong@nudt.edu.cn

Keywords Fracture · Modeling · Animation · Movable cellular automata · Heterogeneous · CUDA

1 Introduction

In the field of computer graphics and virtual reality, people have long been pursuing the realism of computer based simulation of real-world phenomena and scenes. Fracture phenomenon has been one of the focuses of graphics researchers [1], the commonly used modeling methods can be classified into physical-based modeling and geometric-based modeling. Most of the existing methods have a basic assumption that the material must be continuous and isotropic in its material and physical properties, and many simplified physical models have been proposed to reduce the computational cost of simulation. However, when high accuracy is demanded or heterogeneous properties need to be considered, most previous approaches are not suitable due to their limitations.

To simulate fracture phenomena on heterogeneous objects, we propose to use the movable cellular automata (MCA) method. MCA was presented by Russian scholar Professor Psakhie in 1994 at North Carolina State University (NCSU) with Dr. Horie after in-depth study over the cellular automata. The MCA method is suitable for dealing with the nonlinear and large deformation problem. In the field of materials analysis, it has achieved very good results to simulate different materials under different loads.

MCA method discretizes the object into a set of small units with mass in the simulation process at meso-scale, called the movable cellular automata. By studying the relative motion between the automata, we can get the whole law of motion of the object. To deal with the heterogeneous material, we introduce the concept of automata group. The MCA method is used in the traditional mechanical analysis of materials currently. However, the bottleneck of MCA is efficiency. In order to obtain sufficiently accurate simulation results, the objects must be discretized into a large number of automata. However, this will lead to a significant increasing of computational cost, so the traditional CPU-based algorithm is very inefficient. With the demand of the military simulation and computer games, the performance of GPU has increased rapidly in recent years. GPU is designed as a programmable stream processor, which has powerful parallel computing power. It is particularly suitable for general purpose computation-intensive tasks. This paper uses NVIDIA's Compute Unified Device Architecture (CUDA) to compute the interaction between the large number of automata in the MCA method, which greatly improved the computational efficiency.

To summarize, the major contributions of this paper are:

– We introduce the MCA method into computer graphics field to simulate fracture for the first time.
– Different from the previous methods dealing with homogeneous materials, our algorithm can describe the property of heterogeneous material.
– Use the uniform grid data structure for searching the neighboring automata during the MCA simulation.
– All the simulation and rendering run on GPU through vertex buffer object, avoiding the data communication between CPU and GPU.

2 Related works

As a common phenomenon in daily life, fracture simulation has been intensively studied by graphics researchers and material mechanics researchers. In this section, we provide an overview of existing works, which are most related to ours.

2.1 Fracture in graphics

Previous work about fracture simulation in graphics can be categorized into two types: the mesh-based method and meshless method.

Terzopoulos [2] introduced the key concept of using simulation-based methods to animate deformable objects. Based on this, Terzopoulos et al. [3] has done pioneering work in nonelastic deformation and fracture phenomenon in the field of computer graphics. They achieved a relatively simple fracture effect by introducing a simple fracturing mechanism of measuring the instantaneous deformation. In their work, fractures may develop and propagate as the deformation exceeds the elastic limit. Norton et al. [4] used the spring-mass model to model the object that could be broken for the first time. When the distance between two neighboring mass points exceeded the threshold, the connecting spring would be broken. And a fairly complete consideration of the collision detection between fragments was implemented to realize the ceramic teapot fracturing animation. O'Brien et al. [5] utilized the finite element model to analyze the stress and strain, and then took the stress tensors as the fracture criterion. By this method, they obtained quite realistic results while the complex finite element analysis has tremendous computational overhead. Therefore, their method could not achieve real-time performance. Parker et al. [6] inherited the idea of O'Brien et al. [5] and developed a new physical engine named DMM using the simplified finite element analysis (FEA) calculation model; the engine is robust and fast enough for use in the design of game. They showed that their work has been integrated into some popular game platforms successfully. Unfortunately, the engine is too expensive for the common user. Bao et al. [7] also used the FEA method. They computed fracture of a rigid body based on a quasi-static analysis and extended the idea to include ductile deformation. However, such method also faces the problem of tremendous computational cost. Su et al. [8] extended Bao et al.'s work by introducing the kinetic energy and angular momentum conservation during contact and collision processing. With joining the collision center-based preprocess mechanism, they presented a simulation technology for fast and realistic fracture of rigid bodies. All the works above belong to the mesh-based method.

Desbrun and Gascuel [9] introduced the meshless method into the graphics field for the first time. They used a particle system coated with a smooth isosurface for simulating soft inelastic material. Muller et al. [10] presented a meshless animation framework for modeling and simulating elastic, plastic, and melting volumetric objects based on the computation of the discrete displacement field using the Moving Least Squares procedure. For rendering, they used point-based representations for both volume and boundary surface. Later, they presented a geometrically method [11] to simulate deformable point-based objects. Their method did not require preprocessing and provided unconditionally stable dynamics simulations. Pauly et al. [12] presented a meshless animation framework for elastic and plastic materials that fracture. Instead of maintaining a consistent volumetric mesh, they adopted highly dynamic surface and volume sampling method. The limitation is that even very small fragments must be sampled sufficiently dense in order to obtain stable evaluation of the shape functions. Guo and Qin [13] combined the meshless method with modal analysis framework and provided real-time deformation of volumetric objects. Bell et al. [14] presented an effective method for granular material simulation based on particles using Distinct Element Method (DEM). Imagire et al. [1] presented a unified framework for simulating destruction and the generated dust and various sizes of debris based on Extended Distinct Element Method (EDEM). However, the performance of their system is not high enough. Liu et al. [15] introduced a meshless brittle fracture simulation framework based on the Meshless Local Petrov–Galerkin method. They obtained realistic results for simulating fracture of anisotropic materials such as glass or wood. Guo et al. [16] presented a new approach to the physically-based thin-shell simulation of point-sampled geometry via explicit, global conformal point-surface parameterization, and meshless dynamics.

2.2 MCA in material mechanics

MCA method has played an important role in many applications than other numerical algorithms since it originated [17–22]. In recent years, researchers have done a lot of experiments in many different areas using MCA methods. And numerical calculation software based on MCA has been developed to simulate a variety of materials (metal, alloy, ceramic, composite materials, cement, rocks, soil, etc.) under different loads of various mechanical tests (compression, tension, vibration, shear, seismic tests, etc.).

In the rock mechanics, the movable cellular can express geometrical characteristics of rock joints more realistic and handle the nonlinear deformation and failure problems easily, so it is widely used in simulation of landslides and other mechanical process analysis and calculation.

In powder engineering, the MCA method is widely used in the study of complex dynamic of powder in complex environment, as well as in the study of mechanical properties of materials with complicated structure, such as computer-aided design of concrete and accurate fracture process simulation of concrete structures under loads. It can also be used to calculate the corresponding mechanical properties of concrete structures under different loadings. The MCA method for the seismic simulation and its damaging effects also achieved good results.

3 Algorithm overview

Our system consists of two parts: the simulation and rendering. Workflow of the entire system can be described as follows: The 3D model of the object to be simulated is first discretized as a set of discrete automata. When the object is affected by external force, the automata will be activated and the system enter into the MCA simulation stage. After the simulation of each time step, associated automata attributes are updated and directly sent to the GPU for rendering. The overview of our algorithm is illustrated in Fig. 1. Implementation of the algorithm of each time step can be described as follows:

Step 1: MCA-based Simulation

- Step 1.1: Initialize the corresponding automata parameter, including original position and muzzle velocity, etc.
- Step 1.2: Automata is activated by external force and will move to a new position, calculate the new force and displacement for each automata.
- Step 1.3: According to the new position for every automata, estimate the overlapping parameter for every automata pair.
- Step 1.4: Determine whether fracture occurs or not for automata pair based on the fracture criteria.
- Step 1.5: Update every automata after fracture evaluation.

Step 2: Render

- Step 2.1: Read data calculated by CUDA from VBO.
- Step 2.2: Render the data in VBO directly on GPU.

4 MCA method

Movable cellular automata, which combines molecular dynamics and the idea of cellular automata, is a suitable method for dealing with the problem of heterogeneous mechanics. We introduce such method into the field of computer graphics and use it to simulate the fracture of heterogeneous materials.

4.1 Basic concept

Below, we first give some definitions of basic concepts:

Automata: Automata is the smallest unit that composes the object. It is a sphere and has mass and size.

Automata pair: Any two adjacent automata in the simulated object form an automata pair. By the analysis of interactions, positions, and loads of two adjacent automata, the dynamic parameters of the automata pair can be obtained.

Overlapping parameter: It is used to describe the relative position between two neighboring automata, denoted as h^{ij},

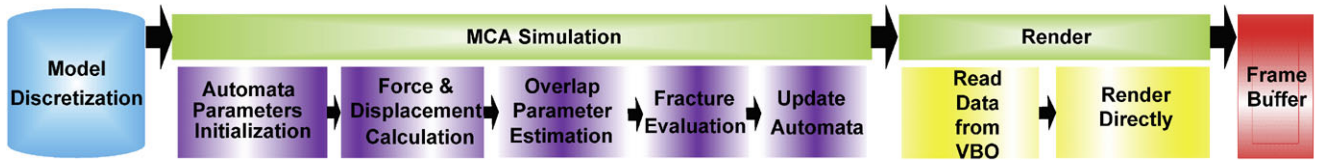


Fig. 1 Algorithm overview for one time step

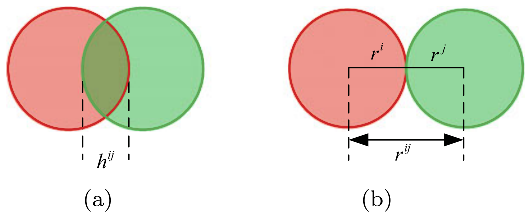


Fig. 2 Automata pair: (**a**) Overlapping (**b**) Contact

which is shown in Fig. 2. And then $h^{ij} = r^{ij} - r_0^{ij}$, where r^{ij} is the real distance from automata i to automata j, $r_0^{ij} = r^i + r^j$, r^i is the radius of automata i, r^j is the radius of automata j.

Automata array: Automata is arranged in 3D space according to certain rules and form the space topology of the simulated object, known as automata array.

Automata group: Automata with the same or similar material mechanism properties form an automata group. So, the object is composed of individual automata and some automata groups from the macroscopical point of view. This means that object is composed on heterogeneous materials.

Automata arrangement: All automata in automata array are organized according to certain rules and constitute the entire object space topology. For the two-dimensional problem, the arrangement of automata arrays can be divided into square arrangement and hexagonal arrangement. According to the different arrangements, the interaction between the automata varies and this will lead to the changes of the equations of motion. In our approach, we assume that automata are six-sided arranged in 3D space. Each automata only interact with the upper, lower, front, rear, left, and right ones to form automata pairs. So, each automata within the automata array is only associated with six "bonds", boundary automata may have less "bonds" according to their positions. And 8 adjacent automata will form a regular hexahedron.

The basic principles of MCA can be described as follows: First, the object must be discretized into a series of small elements of finite size at meso-scale, known as the movable cellular automata. Considering one automata, since it can move, the relative position of each automata in an automata pair will change consequently. Thereby the limitation of traditional finite element method is overcome and the displacement continuity between elements no longer needs to be maintained. Uniting the adjacent automata elements will form the element grid. The interactive state between automata could be changed and form new automata pair with other automata. Consequently, we can link the whole model, and finally all the automata's response function can be integrated as the overall reaction function. The model based on this overall response function is the mechanical model of MCA.

4.2 Automata motion equation

The motion equation of automata is as below:

$$\frac{d^2h^{ij}}{dt^2} = \left(\frac{1}{m^i} + \frac{1}{m^j}\right)p^{ij} + \sum_{k\neq j} C(ij,ik)\psi(\alpha_{ij,ik})\frac{1}{m^i}p^{ik} + \sum_{l\neq i} C(ij,jl)\psi(\alpha_{ij,jl})\frac{1}{m^j}p^{jl} \tag{1}$$

$$\frac{d^2\theta^{ij}}{dt^2} = \left(\frac{q^{ij}}{J^i} + \frac{q^{ji}}{J^j}\right)\tau^{ij} + \sum_{k\neq j} S(ij,jk)\frac{q^ik}{J^i}\tau^{ik} + \sum_{l\neq i} S(ij,jl)\frac{q^jl}{J^j}\tau^{jl} \tag{2}$$

where $(\frac{1}{m^i} + \frac{1}{m^j})p^{ij}$ and $(\frac{q^{ij}}{J^i} + \frac{q^{ji}}{J^j})\tau^{ij}$ are relative acceleration caused by normal force and relative angle acceleration caused by tangential force between automata i and j, $\sum_{k\neq j} C(ij,ik)\psi(\alpha_{ij,ik})\frac{1}{m^i}p^{ik}$ and $\sum_{k\neq j} S(ij,jk)\frac{q^ik}{J^i}\tau^{ik}$ are the force of adjacent automata acted on automata i besides automata j, $\sum_{l\neq i} C(ij,jl)\psi(\alpha_{ij,jl})\frac{1}{m^j}p^{jl}$ and $\sum_{l\neq i} S(ij,jl)\frac{q^jl}{J^j}\tau^{jl}$ are the projection of acceleration that the other adjoint automata acted on automata j besides automata i on the direction from i to j. m is the mass of automata, p is the normal force, $C(ij,ik(jl))$ is associated with the transfer of the parameter h from the pair $ik(jl)$ to the pair ij, $\psi(\alpha_{ij,ik})$ is determined by the Poisson ratio and is related to the mutual arrangement of the pair of elements ij and ik. Here, θ^{ij} is the angle of relative rotation, $q^{ij(jl)}$ is the distance from center of automata $i(j)$ to contact point of automata $j(i)$, τ^{ij} is the pair tangential interaction, and $S(ij,ik,(jl))$ is the certain coefficient associated with transferring the θ parameter from one pair to other.

If $C(ij,ik,(jl)) = 1$ and $S(ij,ik,(jl)) = 1$, the above two motion equations will become a Newton–Euler equation:

$$m^i\frac{d^2r^i}{dt^2} = \sum_j F^{ij} \tag{3}$$

$$\widehat{J^i}\frac{d^2\theta^i}{dt^2} = \sum_j K^{ij} \tag{4}$$

where $F^{ij} = p^{ij} + \tau^{ij}$, $K^{ij} = q^{ij}(n^{ij} \times \tau^{ij})$, n^{ij} is the unit vector and is defined as $n^{ij} = (r^j - r^i)/(q^{ij} + q^{ji})$.

4.3 Constitutive model

In the MCA approach, the mechanical constitutive model of the object consists of two parts: the stress-strain relationship for each automata and the relationship between the interaction force and the relative position for automata pair.

In comparison, the latter has a more significant impact on the constitutive model, which we mainly consider in this paper. We assume that the stress-strain relationship of automata under external loads still follows the continuum mechanics theory of elastic-plastic small deformation. Considering the changes of automata position and connection, the whole object can express more complex constitutive relations, consequently the movable cellular automata can represent different types of material properties, in particular the heterogeneous media.

Let us consider one automata with elastic and plastic changes, the force acting on it from adjoining automata can be regarded as boundary conditions of force and displacement, and the relationship is

$$\sigma_a = \varphi \varepsilon_\alpha + \left(1 - \frac{\varphi}{K}\right)\sigma_c \tag{5}$$

$$\tau_{\alpha\beta} = \frac{\varphi}{2}\gamma_{\alpha\beta} \tag{6}$$

Taking the plane stress state into account, $\alpha, \beta = x, y$; $\sigma_a(\varepsilon_\alpha)$ is the weight on the cross of stress (strain) tensor; $\sigma_c = \frac{\sigma_x+\sigma_y}{2}$, is the average stress; $\tau_{\alpha\beta}(\gamma_{\alpha\beta})$ is the weight of shearing stress (shearing strain); K is the volume module, in isotropy situation, $K = E/(1-2\mu)$, μ is Poisson ratio; and $\varphi = \frac{2}{3}\frac{d\sigma_{\text{int}}(\varepsilon_{\text{int}})}{d\varepsilon_{\text{int}}}$, where $\sigma_{\text{int}} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_x-\sigma_y)^2+\sigma_x^2+\sigma_y^2+6\tau_{xy}^2}$, is the equivalent stress, $\varepsilon_{\text{int}} = \frac{\sqrt{2}}{3}\sqrt{(\varepsilon_x-\varepsilon_y)^2+\varepsilon_x^2+\varepsilon_y^2-\varepsilon_x\varepsilon_y+6\gamma_{xy}^2}$, is the equivalent strain.

The elastic modulus E will be changed with the changes of strain e, and follow the exponential strengthen law or quadratic strengthen law [22], the equations are

$$E = E_0 e^{\alpha e} \quad \text{or} \quad E = E_0\left(1+\alpha e^2\right) \tag{7}$$

where e is the strain, E_0 is the elastic modulus of material, E is the elastic modulus after deformation, and α is the material parameter.

On this basis, we can discuss the influence of changes of relative position and changes of motion state on the constitutive relation. To study any one automata, we assume that automata in a pair are still in contact, which are

$$\varepsilon^{ij} = \frac{h^{ij}}{r_0^{ij}} = \frac{(q^{ij}+q^{ji})-(d^i+d^j)/2}{(d^i+d^j)/2} \tag{8}$$

$$\left(\Delta\varepsilon^{i(j)}+\Delta\varepsilon^{j(i)}\right)\frac{d^i+d^j}{2} = V_y^{ij}\cdot\Delta t \tag{9}$$

$$d^i\Delta\gamma^{i(j)}+d^j\Delta\gamma^{j(i)} = 2V_s^{ij}\cdot\Delta t \tag{10}$$

where ε^{ij} is the strain of automata, Δt is the time step, $\Delta\varepsilon^{i(j)}$ and $\Delta\varepsilon^{j(i)}$ are the increment of strain during Δt, V_y^{ij} is the weight of relative velocity on the center line of two automata, $\Delta\gamma^{i(j)}$ and $\Delta\gamma^{j(i)}$ are the shearing strain increment during Δt, V_s^{ij} is the relative shear velocity, and $V_s^{ij} = \omega_{ef}^{ij}r^{ij} - \omega^i q^{ij} - \omega^j q^{ji}$. ω^i and ω^j are the angular velocity of automata ij, θ_{ef} is the angle between the automata pair center line and x axis, ω_{ef}^{ij} is the angular velocity of θ_{ef}, then $\omega_{ef}^{ij} = V_x^{ij}/r^{ij}$, where V_x^{ij} is the vertical velocity weight.

For the normal force and tangential force between the automata pair, the equations are as below:

$$f_{np}^{ij} = \sigma^{ij}\cdot S^{ij} \tag{11}$$

$$f_{tp}^{ij} = \tau_{np}^{ij}\cdot S^{ij} \tag{12}$$

where f_{np}^{ij} and f_{tp}^{ij} is respectively the normal force and tangential force acted on automata i in automata pair ij, S^{ij} is contact area of automata i, j, and $\sigma^{ij} = \sigma^{ji}$, $\tau^{ij} = \tau^{ji}$, $f_{np}^{ij} = -f_{np}^{ji}$, $f_{tp}^{ij} = -f_{tp}^{ji}$.

The object is composed by a number of automata, each automata is in contact with surrounding automata, a single automata must follow the constitutive equations of continuum mechanics, but the relative distance and motion state for each automata in an automata pair are changing. Using all automata equations together, we can obtain the relationship between outside load and deformation, that is the constitutive relation of the object.

4.4 Fracture criteria

In the MCA, the number of automata is fixed, when the automata is affected by the external force, automata which being compressed or stretched will be activated to obtain a certain velocity and angular velocity. When the critical load is exceeded, the connection between automata will be destructed, leading to fracture phenomenon. In our approach, two failure criteria are used:

- Displacement criteria: Judged by the overlapping parameter between automata. Define $h_{\max}^{ij} = r_0^{ij}\cdot\varepsilon_b$, is the critical value of the overlapping parameter, where ε_b is the strain when the material fractures. Then we can obtain:
  When $h^{ij} < h_{\max}^{ij}$, it is a linked state.
  When $h^{ij} > h_{\max}^{ij}$, it is an unlinked state.
- Equivalent stress criteria: When $\sigma_{\text{int}}^{i(j)} \geq K^{ij}\sigma_b^{ij}$ or $\sigma_{\text{int}}^{j(i)} \geq K^{ij}\sigma_b^{ij}$, it is an unlinked state, where σ_b^{ij} is the strengthened limit of automata $i(j)$, K^{ij} is the viscous coefficient.

5 Implementation

We implement our MCA-based simulation system in CUDA using a uniform grid data structure presented by Simon Green [23]. The calculated results are written to a Vertex Buffer Object (VBO) and are directly rendered on the GPU.

5.1 Heterogeneous modeling

The strength of the meso cell bodies within an object of non-homogeneous material is different. Due to the presence of many locally weak areas, the object presents heterogeneity. Some researchers believe that this nature is the cause of the destruction of heterogeneous materials such as rocks by force. For heterogeneity in the MCA, we define that each cellular automata has different fracture threshold (available in $1, 2, 3, 4, \ldots$ in the probability values). The degree of heterogeneity is expressed by $P(i)$ which is the proportion of the threshold of i ($i = 1, 2, 3, 4, \ldots$) of the total number, and $\sum_{i=1}^{n} P(i) = 1$.

From the physical point of view, the mesoscopic heterogeneity of materials such as rocks refers to different strength and other physical and mechanical properties of particles. In MCA, we use the different initial fracture energy threshold equivalent to represent the material's micro level of heterogeneity. We set fracture energy thresholds for four kinds of cellular automata, marked as 1, 2, 3, and 4. Obviously, the heterogeneity of cellular system containing four fracture energy thresholds is greater than that of the system containing less. In a cellular system containing fixed fracture energy threshold, the smaller the differences in the proportion of the number of cellular energy threshold is, the more uneven the distribution of the cellulars. For instance, in a cellular system containing 4 fracture energy thresholds, the system with $P(1) = P(2) = P(3) = P(4) = 25\ \%$ is more heterogeneous than a system with $P(1) = P(2) = 5\ \%$, $P(3) = P(4) = 45\ \%$. Accordingly, we define three different levels of heterogeneity as follows:

- Low heterogeneous: the proportion of the cellular with the energy threshold $1, 2, 3, 4$ is $0, 0, 10\ \%, 90\ \%$, respectively;
- Medium heterogeneous: the proportion of the cellular with the energy threshold $1, 2, 3, 4$ is $0, 15\ \%, 20\ \%, 65\ \%$, respectively;
- High heterogeneous: the proportion of the cellular with the energy threshold $1, 2, 3, 4$ is $20\ \%, 25\ \%, 25\ \%, 30\ \%$, respectively.

The proportion mentioned above is taken, for example, the actual value must be set according to the actual situation.

5.2 Data structure

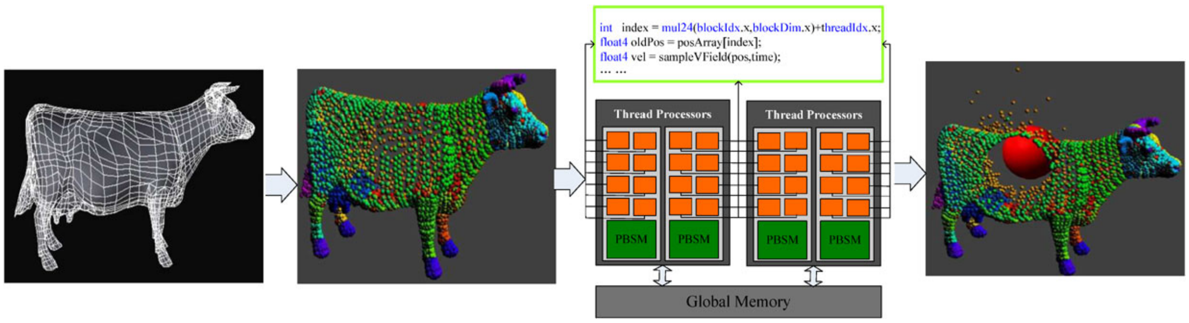
Our main criteria for the choice of an appropriate data structure is the efficiency of the neighborhood search for each automata. A uniform grid could be the simplest spatial subdivision method [23], so we use a three-dimensional uniform grid to reduce the computational burden. We subdivide the simulation space into a grid of uniform cells. The size of each cell is equal to the size of the automata. Each automata is assigned to a specific grid cell according to automata's position. Index of each automata is then stored in a cell. The uniform grid data structure is regenerated for every time step. Due to efficiency requirements, uniform grid allows us to retrieve the adjacent 26 grid cells at most for each automata and take them as the neighbor automata. This makes it possible to perform incremental updates on the GPU.

Let us summarize the two main benefits of the uniform grid used for MCA simulations:

- The construction cost is very low.
- It is easy to access the memory of the grid cell to which an automata belongs.

5.3 Simulation

It has been demonstrated that GPUs can significantly improve the performance of physical simulations [24, 25]. Calculations of individual automata properties are mostly independent, which means that they are well suited for the parallel evaluation on the GPU. The CUDA program consists of host program and kernel program. Host program runs on the CPU in serial mode and kernel program runs on the GPU. The flowchart of CUDA-based computation of automata properties is as shown in Fig. 3. Firstly, the host terminal calculates the required number of threads responsible for the number of automata for *threadNum*, the number of thread blocks for *blocks* and the number of grid for *grids*.



```
int   index = mul24(blockIdx.x,blockDim.x)+threadIdx.x;
float4 oldPos = posArray[index];
float4 vel = sampleVField(pos,time);
··· ···
```

Fig. 3 The automata updating pipeline based on CUDA. The position and velocity datasets of automata are sent to CUDA kernels, which calculate the new position for each automata and send them to rendering process, which draw the automata at their new positions

Then it distributes the array variable space of velocity and position of automata and passes array variable to CUDA. CUDA kernel program calculates the corresponding thread for each automata, and send the results to VBO. The update method of automata properties runs on kernel is our MCA simulation. The thread process of node *u* on the kernel is as follows:

AutomataUpdateKernel (node u)

Step 1: Load *velocity(u)* and *oldPosition(u)* from global memory to shared memory.

Step 2: Synchronize thread to make sure the loading is completed.

Step 3: Calculate the new position of the next moment with MCA.

Step 4: Write the new position to global memory *newPosition(u)*.

5.4 Render

The MCA simulation runs on the GPU using CUDA, and the calculated results need to be rendered. If the results are sent to the CPU, for the communication bandwidth between CPU and GPU is very limited, the data transmission between CPU and GPU will become the bottleneck of the systems. To solve this problem, we store the calculation results of CUDA in the vertex buffer object, and render directly on the GPU at the end of each time step.

Here, we describe our rendering approach in detail. First, we take the positions of automata as vertex attributes in GPU, stored with OpenGL vertex buffer objects. OpenGL buffer objects can be mapped into the address space of CUDA, so that CUDA can read data from OpenGL or write data to OpenGL. The mapping process of CUDA to VBO consists of three steps: object registration, read or write, and free the map. The registration can simply be done with a function call:

GLuint vbo;
cudaGLRegisterBufferObject(vbo);

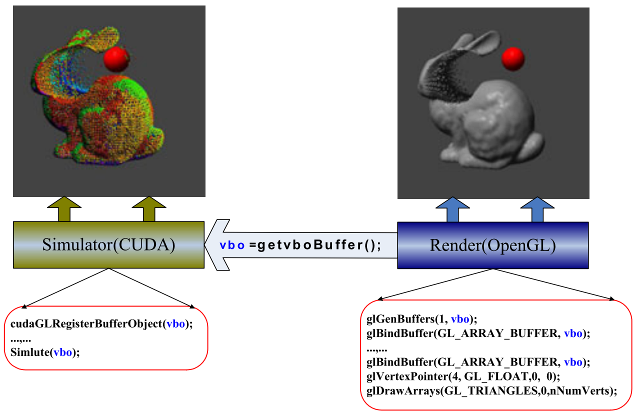
After registration, the kernel can read from or write to vertex buffer using the device storage address returned by *cudaGLMapBufferObject()*:

*float *oldPos*;
*cudaGLMapBufferObject((void**)& oldPos, vbo)*;

The map freeing is completed by *cudaGLUnmapBufferObject()* and the unregister can be completed by *cudaGLUnregisterBufferObject()*, the code is as below:

cudaGLUnmapBufferObject(vbo);
cudaGLUnregisterBufferObject(vbo);

Our models are represented as a set of automata in the MCA simulation. After simulation, we render a triangle for



```
vbo =getvboBuffer();
```

```
cudaGLRegisterBufferObject(vbo);
...,...
Simlute(vbo);
```

```
glGenBuffers(1, vbo);
glBindBuffer(GL_ARRAY_BUFFER, vbo);
...,...
glBindBuffer(GL_ARRAY_BUFFER, vbo);
glVertexPointer(4, GL_FLOAT,0,  0);
glDrawArrays(GL_TRIANGLES,0,nNumVerts);
```

Fig. 4 VBO data transmission between CPU and GPU

each automata. The specific process can be described as follows: First, calculate the center of mass for each triangle of the object model. And then store the offset of the vertex of triangle to the center of mass, which is mapped to a vertex buffer object VBO. Then the VBO is obtained by simulator by using a function *getvboBuffer()*, as shown in Fig. 4. The simulator registers the VBO and assigns one automata at the center of mass to use it to perform the MCA simulation. After simulation, based on the new position of mass center and the precalculated offset, we can obtain the new vertex positions. At last, we render new triangles with the new calculated vertex positions for each automata.

6 Results

We implement our algorithm on a GPU with NVIDIA Quadro FX 4600 graphics card and an Intel(R) Core(TM) i7 CPU 920@2.67 GHz with 4.0 GB RAM. The development environment is Visual C++.Net 2008.

6.1 Effectiveness

We first take two relatively simple models to validate the effectiveness of our system.

The cow model has 2903 vertices(see Fig. 5), in the MCA simulation, the model is discretized into 5804 automata. The red ball is considered to be a completely rigid sphere. Given the mass of the red ball with a certain initial velocity, it collided with the cow model. The intervention of external forces will make the interaction between cellular automata change, resulting in displacement of the cellular automata, so that the interposition relationship of the cellular automata changes. The automata affected by the hit of the ball is separated from the surrounding automata, starts moving at a certain speed, and flies outward turning into independent fragments. The results are shown in Fig. 5. Figure 5(a) illustrates how a cow model is represented by discontinuous

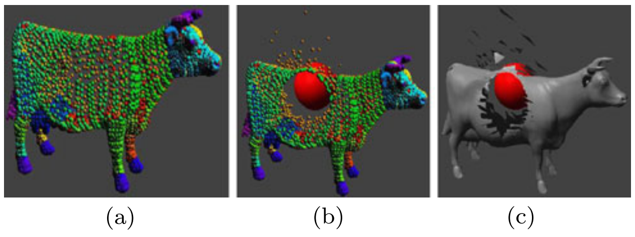


Fig. 5 Fracture of cow

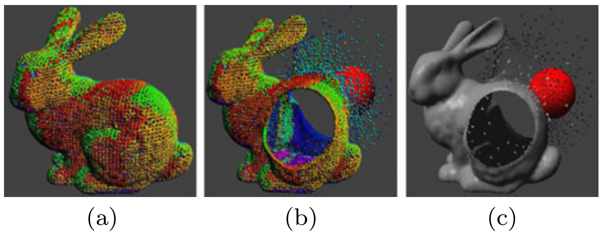


Fig. 6 Fracture of bunny

automata. Figure 5(b) demonstrates the fracture process of cow by discrete automata. Figure 5(c) shows the rendering results. Most automata in Fig. 5(a) are green, which have the same material property. According to the definition mentioned in Sect. 5.1, we can find that the cow model belongs to be low heterogeneous.

To evaluate the effectiveness of our algorithm on heterogeneous objects, we use the Stanford Bunny model, which is composed of different material. The model is discretized into 16301 automata where different color indicate different material. From Fig. 6(a), we can see that most automata are yellow, green, or red. The automata with the same color have the same fracture energy threshold. So, Bunny is a medium heterogeneous model. Figure 6(b) demonstrates the fracture process of bunny by discrete automata. Figure 6(c) shows the rendering results.

6.2 Robustness

Taking into account the powerful parallel computing capability of GPU and to testify the efficiency and robustness of proposed MCA simulation algorithm based on CUDA, we use larger and more complex models for experiment.

Venus model has 19755 vertices and is discretized into 43357 automata. Figure 7(a) illustrates how a Venus model is represented by automata. For simplicity, the Venus is defined to be low heterogeneous. So, we can see from Fig. 7(a) that the upper body of Venus is mainly composed of red automata while the lower body of Venus is composed of blue ones. Figure 7(b) demonstrates the fracture process of Venus by discrete automata. The blue automata has the lower fracture energy threshold than the red ones, so we can find from Fig. 7(b) that more fragments are formed from the lower body of Venus. Figure 7(c) shows the rendering results.

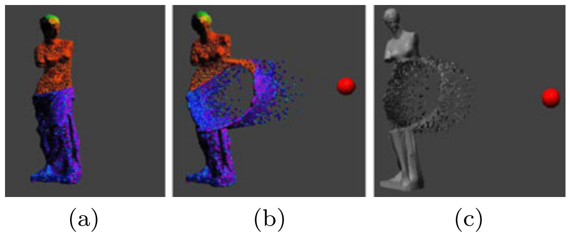


Fig. 7 Fracture of venus

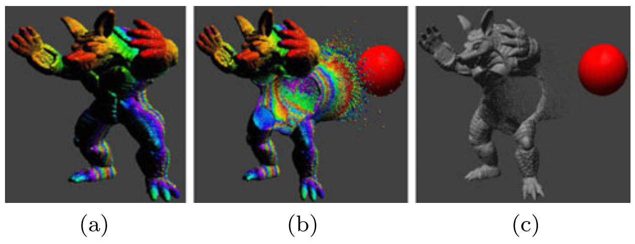


Fig. 8 Fracture of armadillo

Table 1 System performance

Model	Vertexes	Automata	Average FPS	
			Render particles	Render triangles
Cow	2903	5804	170.6	62.3
Bunny	8146	16301	145.1	60.6
Venus	19755	43357	99.6	27.9
Armadillo	172974	345944	30.1	15.0

The Armadillo model has 172974 vertices, and a total of 345944 automata are involved in the MCA simulation. For the armadillo model is very fine with hundreds of thousands of vertices and faces, the simulation and rendering results are much more detailed. We can observe some minor deformation and small cracks during the fracture process, and after fracture a number of small debris can also be included in the results. Being the most complex model, the armadillo should be defined to high heterogeneous obviously, as shown in Fig. 8(a). A variety of colors of automata are distributed on the armadillo model where different colors also indicate different material property. Figure 8(b) demonstrates the fracture process of armadillo by discrete automata. Figure 8(c) shows our rendering results.

6.3 Performance analysis and comparison

We have done several experiments on different models, the simulation and rendering performances are shown in Table 1. The Cow model used 5804 automata during the MCA simulation, and the average frame rate achieved 170.6 fps with particles for rendering; when to render the triangle faces, the frame rate dropped to 62.3 fps. The Bunny model has 16301 automata for MCA simulation, the average frame rate is 145.1 fps with particles for rendering and 60.6 fps with triangle faces. The Venus model used 43357 automata for simulation, and the average fps is 99.6 and 27.9, respectively. Armadillo is used for the most complex models, 345944 automata were involved in the MCA simulation. With a increase of the calculation, the simulation efficiency is still maintained at 30.1 fps or so, and with triangle faces for rendering the average frame rate dropped to 15.0 fps.

Imagire et al. [1] adopted the discrete element method for fracture animation, in their method the model is discretized into a series of discrete particles. The basic idea is similar to ours, but the efficiency of their algorithm is not very high. Table 2 is the performance comparison between our method and Imagire et al.'s. Their experimental machine is configured to Intel Core 2 Extreme X6800 2.93 GHz and NVIDIA GeForce 8800 GTX, but when the number of elements in the system was up to 2048, the frame rate was only 9.1 fps. In our system, the number of automata in armadillo model has achieved 345944, the performance of simulation is still up to 30.1 fps. And in Imagire's paper, only the plate or cylinder model is given for example. Our approach can deal with a more complex model, and the heterogeneity of the material is taken into account, so it is more general.

As we all know, concrete is a typical heterogeneous material. Based on this, we compare our algorithm with several previous methods, where we simulate the fracture process of a concrete wall hit by a steel ball. Figure 9 is the simulation results comparisons chart, Figs. 9(a)–9(b) are the results from literature [1], Figs. 9(c)–9(d) are produced by literature [7] and Figs. 9(e)–9(f) are our results. The literature [1] used a particle-based method which is similar to us and the performance is up to 8.8 fps. The literature [7] is based on finite element method and the efficiency is about 40 seconds/frame. Our concrete wall in Figs. 9(e)–9(f) consists of over 3000 automata during simulation and the rendering performance of our system is more than 180 fps, while the effect of our realization is consistent with realistic. In Figs. 9(a)–9(f), there is only one concrete wall, and in order to verify system performance we put four concrete walls side by side in Figs. 9(g)–9(h). In this scene, we use more than 12000 automata during simulation. Even so, the frame rate in Figs. 9(g)–9(h) is still more than 30 fps.

The experimental results show that our simulation framework is stable and efficient.

Table 2 Performance comparison

	Max number of elements	Average FPS	
		without rendering	with rendering
Imagire 09	2048	9.1	8.8
Our method	345944	30.1	15.0

7 Conclusion

In this paper, we have introduced a novel approach for simulating the fracture phenomenon. Different from the previous method, our approach can handle the fracture of heterogeneous materials. During the MCA simulation, we presented to adopt the concept of automata group for the exhibition of

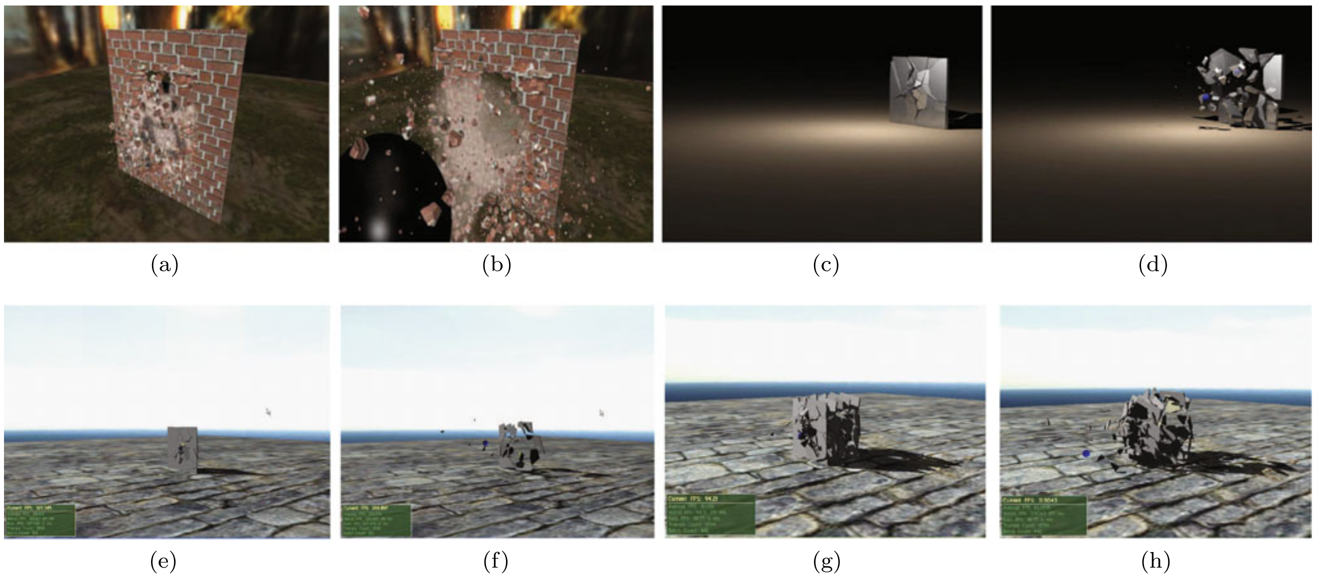


Fig. 9 Concrete wall fracture result comparison: (**a**) and (**b**) are courtesy of the authors of Dr. Takashi Imagire et al., (**c**) and (**d**) are courtesy of the authors of Dr. Zhaosheng Bao et al.

heterogeneity of objects. And for the elastic modulus of constitutive model, the exponential strengthen law and quadratic strengthen law are used to adapt the impacting effect. Displacement criteria and equivalent stress criteria are used as the fracture criteria. With this approach, we have obtained realistic experimental results.

In order to avoid the unnecessary and redundant communication between CPU and GPU, we use the vertex buffer object to directly render the results calculated by CUDA. Profiting from the powerful computational capability of GPU, the performance of our system can meet the real-time requirement.

In our current approach, we only consider the physical and material mechanical properties for automata, which is often not sufficient in simulating real world objects. In addition to deformation and motion caused by external force, the automata is also affected by the environmental temperature, etc. Especially in the simulation of fracture caused by explosion or shock wave, we must consider the release heat by complex chemical reactions and blast effects on automata properties. The current MCA method can be further improved to take these issues into account.

Acknowledgements We first thank the anonymous reviewers for their careful and valuable comments. We are grateful to Aiping Wang and Kai Xu from NUDT for fruitful discussions. This work is supported by the National Basic Research Program (No. 2009CB723803) and the National Science Foundation Program (No. 60873120) of China.

References

1. Imagire, T., Johan, H., Nishita, T.: A fast method for simulating destruction and the generated dust and debris. Vis. Comput. **25**, 719–727 (2009)
2. Terzopoulos, D., Witkin, A.: Physically based models with rigid and deformable components. IEEE Comput. Graph. Appl. **8**, 41–51 (1988)
3. Terzopoulos, D., Fleischer, K.: Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, pp. 269–278 (1988)
4. Norton, A., Turk, G., Bacon, B., Gerth, J., Sweeney, P.: Animation of fracture by physical modeling. Vis. Comput. **7**, 210–219 (1991)
5. O'Brien, J.F., Hodgins, J.K.: Graphical modeling and animation of brittle fracture. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 137–146 (1999)
6. Parker, E.G., O'Brien, J.F.: Real-time deformation and fracture in a game environment. In: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, New Orleans, Louisiana, pp. 165–175 (2009)
7. Bao, Z., Hong, J., Teran, J., Fedkiw, R.: Fracturing rigid materials. IEEE Trans. Vis. Comput. Graph. **13**, 370–378 (2007)
8. Su, J., Schroeder, C., Fedkiw, R.: Energy stability and fracture for frame rate rigid body simulations. In: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, New Orleans, Louisiana, pp. 155–164 (2009)
9. Desbrun, M., Cani-Gascue, M.: Animating soft substances with implicit surfaces. In: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, pp. 287–290 (1995)
10. Muller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., Alexa, M.: Point based animation of elastic, plastic and melting objects. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Grenoble, France, pp. 141–151 (2004)
11. Muller, M., Heidelberger, B., Teschner, M., Gross, M.: Meshless deformations based on shape matching. In: Proceedings of the ACM SIGGRAPH 2005 Papers, Los Angeles, California, pp. 471–478 (2005)
12. Pauly, M., Keiser, R., Adams, B., Dutre, P., Gross, M., Guibas, L.J.: Meshless animation of fracturing solids. In: Proceedings of the ACM SIGGRAPH 2005 Papers, Los Angeles, California, pp. 957–964 (2005)
13. Guo, X., Qin, H.: Real-time meshless deformation. Comput. Animat. Virtual Worlds **16**, 189–200 (2005)
14. Bell, N., Yu, Y., Mucha, P.J.: Particle-based simulation of granular materials. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Los Angeles, California, pp. 77–86 (2005)
15. Liu, N., He, X., Li, S., Wang, G.: Meshless simulation of brittle fracture. Comput. Animat. Virtual Worlds **22**, 115–124 (2011)
16. Guo, X., Li, X., Bao, Y., Gu, X., Qin, H.: Meshless thin-shell simulation based on global conformal parameterization. IEEE Trans. Vis. Comput. Graph. **12**(3), 375–385 (2006)
17. Psakhie, S.G., Horie, Y., Korostelev, S.Y., Smolin, A.Y., Dmitriev, A.I., Shilko, E.V., Alekseev, S.V.: Method of movable cellular automata as a tool for simulation within the framework of mesomechanics. Russ. Phys. J. **38**, 1157–1168 (1995)
18. Psakhie, S.G., Korostelev, S.Y., Smolin, A.Y., Dmitriev, A.I., Shilko, E.V., Moiseyenko, D.D., et al.: Movable cellular automata method as a tool for physical mesomechanics of materials. Phys. Mesomech. **1**(1), 89–101 (1998)
19. Psakhie, S.G., Moiseyenko, D.D., Smolin, A.Y., Shilko, E.V., Dmitriev, A.I., Korostelev, S.Y., et al.: The features of fracture of heterogeneous materials and frame structures. Potentialities of mca design. Comput. Mater. Sci. **16**(1–4), 333–343 (1999)
20. Psakhie, S.G., Zavshekand, S., Jezershek, J., Shilko, E.V., Dmitriev, A.I., Smolin, A.Y., et al.: Computer-aided examination and forecast of strength properties of heterogeneous coal-beds. Comput. Mater. Sci. **19**(1–4), 69–76 (2000)
21. Psakhie, S.G., Horie, Y., Ostermeyer, G.P., Korostelev, S.Y., Smolin, A.Y., Shilko, E.V., et al.: Movable cellular automata method for simulating materials with mesostructure. Theor. Appl. Fract. Mech. **37**(1–3), 311–334 (2001)
22. Chen, K.: MCA method and the study of penetration of projectile into concrete target. Ph.D. Thesis, Nanjing University of Science and Technology, Nanjing (December 2005)
23. Green, S.: CUDA particles. White paper (November 2007)
24. Georgii, J., Westermann, R.: Mass-spring systems on the gpu. Simul. Model. Pract. Theory **13**(8), 693–702 (2005)
25. Liu, W., Schmidt, B., Voss, G., Mller-Wittig, W.: Accelerating molecular dynamics simulations using graphics processing units with cuda. Comput. Phys. Commun. **179**(9), 634–641 (2008)

Jiangfan Ning received his M.Sc. degree from the National University of Defense Technology in 2005. Currently he is a Ph.D. candidate at the School of Computer Science of the National University of Defense Technology. His research interests include virtual reality and computer animation.

Huaxun Xu received his Ph.D. in Computer Science from the National University of Defense Technology in 2011. Now he is a researcher in Army Aviation Institute of PLA, China. His current research interests include scientific visualization and virtual reality technologies.

Bo Wu received his M.Sc. degree in computer science and technology from National University of Defense Technology China in 2012. Now he is a PhD candidate in computer graphics group at Institute of Science and Technology Austria. His research interests include computer graphics, solid and fluid simulation.

Liang Zeng received his Ph.D. in Computer Science from the National University of Defense Technology. After postdoctoral study, he is currently a professor at the National University of Defense Technology. He is involved in several National High Technology Research and Development programs and the National Basic Research programs of China. He is mainly engaged in research and development of virtual reality and visualization.

Sikun Li is a professor at the School of Computer Science of the National University of Defense Technology. He is the principal of many national research projects of China such as the National Science Foundation program, the National High Technology Research and Development program and the National Basic Research program. His research interests include scientific visualization, virtual reality, very large scale integration, and the design of SoC.

Yueshan Xiong is a professor at the School of Computer Science of the National University of Defense Technology. He is involved in several National High Technology Research and Development programs and the National Basic Research programs of China. His research interests include virtual reality and computer graphics.