Realtime Simulation and Rendering of Dynamic and Complex Fracture Phenomena

Category: Research

Abstract

We present a novel framework for realtime simulation and rendering of the breaking and fracturing of elastic objects. First, we use the linearized forms of the equations of elasticity in a novel multiresolution 'ghosting' framework for fast simulation of the deformation and breaking of a tetrahedral mesh on a CPU. This is coupled with a fast interactive subsurface scattering method suitable for complex fracture scenes, which we implemented on a GPU. We illustrate our method for large volumetric models.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Animation;

Keywords: finite elements, fracture

1 Introduction and Related Works

Breaking and fracturing of elastic objects is of great interest to Computer Graphics, particularly the gaming and movie industry. The increasng demand of fast and robust simulation of objects in destruction is vital in interactive games and modelling applications. It is also an extremely useful tool for people working in the field of demolition, or crash analysis. We have developed a complete framework for simulating and rendering fracture phenomena of unprecendented complexity. This is achieved by a fast linearized scheme implemented on the cpu coupled with a fast subsurface scattering method implemented on the GPU.

Deformable models was first discussed in the graphics literature in [Terzopoulos et al. 1987] Other early work on finite element modeling includes [Gourret et al. 1989; Chen and Zeltzer 1992]. Some of the more recent work includes the adaptive framework of [Debunne et al. 2001], the rotation based approach in [Muller et al. 2002], and the finite volume muscle models of [Teran et al. 2003].

Most graphics researchers simply break connections or springs between elements when the force is high for simulation of fracture, see e.g. [Norton et al. 1991; Hirota et al. 1998; Mazarak et al. 1999; Smith et al. 2001]. Interesting two dimensional results were obtained in [Neff and Fiume 1999] in the context of blast waves. [Muller et al. 2001] treated objects as rigid bodies between collisions, and used static finite element analysis techniques during collisions. They used the principal stress components to separate tetrahedra occasionally refining large tetrahedra before splitting along element boundaries. Similarly, [Muller and Gross 2004; Muller et al. 2004] fracture between element boundaries in a FFD framework and maintain a watertight embedded surface mesh, however their fracture surfaces are quite limited by the coarse simulation mesh. The state of the art in fracture for computer graphics is the work of [O'Brien and Hodgins 1999; Yngve et al. 2000; O'Brien et al. 2002] which used a pseudo principal stress and continuous remeshing.

1.1 Elasticity, Energies, Equation of Motion

Let the point $\mathbf{x} = [x_1, x_2, x_3]^T$ in the un-deformed coordinates, and let $\mathbf{p}(\mathbf{x}) = \mathbf{x} + \mathbf{u}(\mathbf{x})$ in the deformed coordinates. Now we will introduce the notion of stress and strain. Stress measures the force applied perpendicular to a surface element ΔA. Hence the force on the element is $f = \Delta A \sigma \hat{n}$ where $\hat{n}$ is the normal to the surface element. The strain is a measurement of the distortion present in the body. In 1D, the strain ε is simply the change in length perpendicular to ΔA divided by the original length, and the relationship between stress and strain can be simply stated as $\sigma = E\varepsilon$ where E is the elasticity Modulus, a scalar in this case. In 3D, a common measure of strain is the Green's strain tensor, which is invariant to rotation and translation. It is the 3x3 matrix

Figure 1: Shattering of a torus after it collides with the ground.

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial \mathbf{p}}{\partial x_i} + \frac{\partial \mathbf{p}}{\partial x_j} - \delta_{ij}\right) \tag{1}$$

Noting that

$$\frac{\partial p_k}{\partial x_i} = \frac{\partial x_k}{\partial x_i} + \frac{\partial u_k}{\partial x_i} = \frac{\partial u_k}{\partial x_i} + \delta_{ik}$$

we have,

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \frac{\partial \mathbf{u}}{\partial x_i} \cdot \frac{\partial \mathbf{u}}{\partial x_i}\right) \tag{2}$$

Here the diagonal terms of the strain tensor represent deformation in the normal direction, while the off diagonal terms represent shearing. The stress tensor can be computed from the strain using an extension of Hooke's law,

$$\sigma_{ij} = 2G\left(\frac{v}{1-2v} tr(\varepsilon)\delta_{ij} + \varepsilon_{ij}\right) \tag{3}$$

Here the constant G stands for the shear modulus, which determines how much the object resists deformations. The coefficient v is called poisson's ratio. It determines the extent to which strains in one direction are related perpendicular to it, and is hence related to volume preservation. The elastic potential energy $U(\mathbf{x})$ is the amount of energy stored in the object in a deformed state. It can be calculated by integrating the energy density over the entire region Ω of the object, giving

$$U = G\int_\Omega \left(\frac{v}{1-2v} tr^2(\varepsilon)\delta_{ij} + \varepsilon_{ik}\varepsilon_{jl}\delta_{ik}\delta_{jl}\right) d\Omega \tag{4}$$

Let $\rho(x)$ be the mass density in the body. The kinetic energy, a simple generalization of $mv^2/2$ in 1D, is simply

$$T = \frac{1}{2}\int_\Omega \rho(x)\frac{d\mathbf{p}}{dt} \cdot \frac{\partial \mathbf{p}}{\partial t}\, dV \tag{5}$$

Now we can write down the equations of Motion, which takes the Euler-Langrange form.

$$\frac{d}{dt}\left(\frac{\partial T(\dot{\mathbf{u}})}{\partial \dot{\mathbf{u}}}\right)+\frac{\partial U(\mathbf{u})}{\partial \mathbf{u}}+C\dot{\mathbf{u}}=\mathbf{f_{ext}} \tag{6}$$

where $\mathbf{f_{ext}}$ is the net external force. Define the mass matrix to be $\mathbf{M}=\int_\Omega \rho(x)dV$, then substituting equation (5) into (6) gives us

$$\mathbf{M}\ddot{\mathbf{u}}+C\dot{\mathbf{u}}+\frac{\partial U}{\partial \mathbf{u}}=\mathbf{f_{ext}} \tag{7}$$

1.2 Hierarchical Basis

We use a tri-linear basis functions in our multiresolution framework, similar to Capell, et al[Capell et al. 2002b] and [Capell et al. 2002a]. Let $\phi^a(\mathbf{x})$ be elements of the hierachical basis $B=\{\phi_a\}$, expressing each point on the body in this basis, gives $\mathbf{p}=\mathbf{x}+\mathbf{u}=\sum_{\mathbf{a}}(\mathbf{y}+\mathbf{w})\phi_a$ where $\mathbf{x}=\sum_{\mathbf{a}}\mathbf{y}\phi_a$ and $\mathbf{u}=\sum_{\mathbf{a}}\mathbf{w}\phi_a$. At the top level the basis functions correspond to the original control lattice L. After the i^{th} subdivision, we get the complex L_i, which gives the basis B_i. Substituting into equations (6) and (7), we get

$$\frac{d}{dt}\left(\frac{\partial T(\dot{\mathbf{w}})}{\partial \dot{\mathbf{w}}}\right)+\frac{\partial U(\mathbf{w})}{\partial \mathbf{w}}+C\dot{\mathbf{w}}=\mathbf{M^{ab}}\ddot{\mathbf{w}}+\frac{\partial U(\mathbf{w})}{\partial \mathbf{w}}+C\dot{\mathbf{w}}=\mathbf{f_{ext}} \tag{8}$$

Consequently, the new mass matrix becomes

$$\mathbf{M^{ab}}=\int_\Omega \rho(x)\phi^a\phi^b dV \tag{9}$$

The major advantage of this hierarchical formulation is that we can adaptively choose the sub-basis B_i of B, such that detail is added where-ever it is needed.

1.3 Linearization and Equation Solving

Although $\mathbf{M}$, $\mathbf{C}$, and $\frac{\partial U}{\partial \mathbf{u}}$ are large sparse matrices, their evaluation takes too long for interactive applications. First, in order to resolve $\ddot{\mathbf{x}}$, $\mathbf{M^{-1}}$ must be solved at the beginning of every time step, which is impractical for interactive techniques. Some simplifying assumptions are needed. We assume that the damping term C is a simple diagonal matrix, and we can in fact rewrite it as μI. In the hierarchical framework, it first subdivide the control mesh to a desired level and compute basis function values at each vertex, then it tetrahedralize the domain (allows piecewise linear approximation of functions) and perform the numerical integrate over each tetrahedron using linear approximations of basis functions. Now, let $\mathbf{F}(\mathbf{u})=\frac{\partial U(\mathbf{u})}{\partial \mathbf{u}}$ and substituting $\mathbf{u}=\mathbf{p}-\mathbf{x}$ into (7), we get

$$\mathbf{M}\ddot{\mathbf{p}}+\mathbf{C}\dot{\mathbf{p}}+\mathbf{F}(\mathbf{p}-\mathbf{x})=\mathbf{f_{ext}} \tag{10}$$

We start out with the simplifying assumption that $\mathbf{F}$ is a linear function of $\mathbf{u}=\mathbf{p}-\mathbf{x}$, that is a linear strain model from the beginning, resulting in simpler and more intuitive equations that can be more readily used for plasticity and fracturing. We can derive this linear strain model from the non-linear model as follows. We must first linearize the strain tensor, by making the approximation

$$\varepsilon_{ij}\approx\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right) \tag{11}$$

Now we can rewrite the six distinct terms of this symmetrical tensor in vector form for simplicity. Let,

$$e=\begin{pmatrix}\varepsilon_{11} & \varepsilon_{22} & \varepsilon_{33} & \varepsilon_{12} & \varepsilon_{23} & \varepsilon_{13}\end{pmatrix}^T=\mathbf{P}\mathbf{u}(t) \tag{12}$$

where $\mathbf{P}$ is the resulting operator matrix,

$$\mathbf{P}=\begin{pmatrix}\frac{\partial}{\partial x_1} & 0 & 0\\ 0 & \frac{\partial}{\partial x_2} & 0\\ 0 & 0 & \frac{\partial}{\partial x_3}\\ \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_1} & 0\\ 0 & \frac{\partial}{\partial x_3} & \frac{\partial}{\partial x_2}\\ \frac{\partial}{\partial x_3} & 0 & \frac{\partial}{\partial x_1}\end{pmatrix} \tag{13}$$

As noted before, e_{11}, e_{22}, e_{33} is the strain in the normal, and $e_{12}=e_{21}$, $e_{23}=e_{32}$, $e_{31}=e_{13}$ is the shear strain. Now, we can write out equation (3) explicitly. Let

$$\gamma=1+\frac{\nu}{1-2\nu}\quad \Pi_3=\begin{pmatrix}0&1&1\\0&0&1\\0&0&0\end{pmatrix}$$

then, the strain tensor in vector form works out to be

$$\sigma = \begin{pmatrix}\sigma_{11} & \sigma_{22} & \sigma_{33} & \sigma_{12} & \sigma_{23} & \sigma_{31}\end{pmatrix}^T \tag{14}$$

$$= 2G\begin{pmatrix}\gamma I_3+(1+\gamma)\Pi_3 & \mathbf{0}\\ \mathbf{0} & I_3\end{pmatrix}e=\mathbf{E}e \tag{15}$$

Now we can discretize over each tetrahedral element j. Let $\hat{\mathbf{u}}$ be the linear displacement vectors of the coordinates of the four vertices of the tetrahedral element (hence, it has 12 elements). Let $\mathbf{u}(\mathbf{x})=\mathbf{H_j}\cdot\hat{\mathbf{u}}$, where $\mathbf{H_j}$ specify the original shape of the tetrahedron, then we can compute the potential energy of each element from equation (4), which simplifies to

$$U_j=\int_v\frac{1}{2}(e^T\cdot\sigma)dV=\frac{1}{2}V_j(\mathbf{PH_j}\cdot\hat{\mathbf{u}})^{\mathbf{T}}(\mathbf{E}\cdot\mathbf{PH_j}\cdot\hat{\mathbf{u}}) \tag{16}$$

$$=\frac{1}{2}V_j\hat{\mathbf{u}}^{\mathbf{T}}(\mathbf{PH_j})^{\mathbf{T}}\mathbf{E}(\mathbf{PH_j})\hat{\mathbf{u}} \tag{17}$$

Hence if we let $\mathbf{K_j}=V_j(\mathbf{PH_j})^{\mathbf{T}}\mathbf{E}(\mathbf{PH_j})$, then

$$\mathbf{K_j}\hat{\mathbf{u}}=\frac{\partial U_j(\hat{\mathbf{u}})}{\partial\hat{\mathbf{u}}}=\mathbf{F}(\hat{\mathbf{u}}) \tag{18}$$

This gives us the total elastic force on a tetrahedron element. The net force on each node, is the sum of forces from all adjacent tetrahedral. Therefor if we let $\mathbf{F}(\mathbf{u})=\mathbf{F}(\mathbf{p}-\mathbf{x})=\mathbf{K}\cdot(\mathbf{p}-\mathbf{x})$, where $\mathbf{x}$ and $\mathbf{p}$ are $3n$ vectors, containing the coordinates of the n undeformed and deformed points, then we can form the $3nx3n$ stiffness matrix $\mathbf{K}$ by $\mathbf{K}=\sum_{\mathbf{j}}\mathbf{K_j'}$, where $\mathbf{K_j'}$ can be imagined to be $\mathbf{K_j}$ expanded to the size of $\mathbf{K}$ and padded by zeros at coordinates of vertices not adjacent to it. Hence, we have the following set of linear ODE's,

$$\mathbf{M}\ddot{\mathbf{p}}+C\dot{\mathbf{p}}+\mathbf{K}(\mathbf{p}-\mathbf{x})=\mathbf{f_{ext}} \tag{19}$$

This is can be easily solved by the implicit Euler scheme,

$$(\mathbf{M}+\Delta tC+\Delta t^2\mathbf{K})v^{i+1}=\mathbf{M}v_i-\Delta t(\mathbf{K}\mathbf{p}^i-\mathbf{f_{ext}})$$
$$\Delta\mathbf{p}=\Delta t\mathbf{v}^{i+1}$$

Linearized forces, are only accurate close to the equilibrium condition, since it does not preserve volume under larger deformations (we threw out the term that does), and is not invariant to rigid body transformations. This assumption is valid in our scenario, where objects are assumed to be fairly stiff.

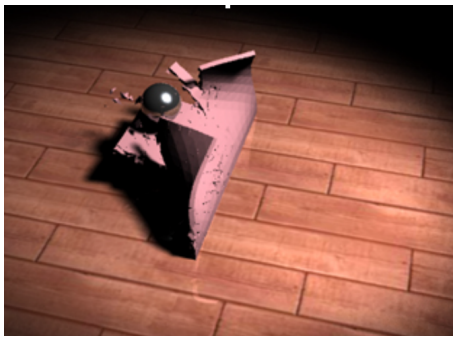

Figure 2: Large stress causes an elastic wall to fail.

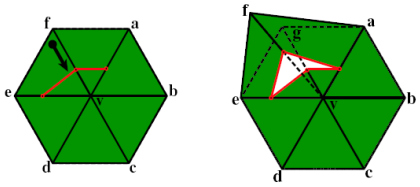

Figure 3: Here, node *a* is scooped out of the one ring of node *b*, and thus *b* donates a virtual copy of itself (node *e*) to node *a*. This creates the degrees of freedom needed for the crack to open.

1.4 Criteria for Fracturing

A material can only withstand a limited amount of stress, before it fractures. The maximum stress criteria occurs when the largest eigenvalue of the stress tensor exceeds a prescribed amount d_max. We can rewrite it in symmetrical tensor form, and diagonalize it.

$$\begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{31} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{23} & \sigma_{33} \end{pmatrix} = \mathbf{M}^T Diag(\tau_1, \tau_2, \tau_3)\mathbf{M}$$

Where τ_1, τ_2, τ_3 are the eigenvalues, and we simply take $\tau_{max} = Max(\tau_1, \tau_2, \tau_3)$. Also compute the eigenvector n_{max} of this eigenvalue, which represents the direction of fracture. If $\tau_{max} > d_{max}$ at an element of a tetrahedron, then fracture occurs there.

1.5 Ghost Element Method

As an object breaks, a method for remeshing it is required. In order to split the mesh, we use the "virtual node algorithm" remeshing method proposed in [Molino et al. 2004] where we represent the fragmented surface as embedded in the simulation mesh. When the material fractures through an element, the element is replicated and each replica is simulated as partially void. This avoids global remeshing, takes advantage of the time coherence of the simulation mesh, and permits a possibly branched piecewise linear fracture path without sliver triangles. This virtual node algorithm considers scoops out of the one rings of each triangle node (i.e. regions where the segmentation boundary completely separates a node from some of its neighbors). Subsequently, for each distinct scoop out of the one ring of a node *v*, a virtual copy of *v* is created and donated to the group of nodes within the given scoop giving the mesh the degrees of freedom it needs to break apart. Figures 3 shows an example of this method in action in 2D.

2 Realtime Subsurface Scattering of Fracture Phenomena

For realtime rendering of fracture phenomena on a GPU, we propose a simplified discrete importance sampling method for rendering complex deformable objects. Russian roulette technique can be applied for ray tracing terminations. The subsurface scattering algorithm is tested in a complex geometry and some restrictions on applying dipole diffuse approximations are given. The scenes for fracturing animation can be quite complicated: thousands of object pieces with high specular reflections and refractions, using materials like the glass, for example. The general ray tracing algorithm cannot work properly because the tracing depth can become extremely deep before termination, sometimes more than 20, even though we only consider specular reflections and refractions. Each ray sample needs to propagate about 220 new rays, while most of them provide only a little contribution to the whole outgoing radiance. [Hall 1983] suggested just abandoning those rays with estimated intensities less than some threshold. But the major problem is that the summed errors can be significant even though the individual intensity is unnoticeable. Instead, we use a Russian Roulette method.

2.1 Subsurface Scattering

As described in [Jensen et al. 2002], subsurface scattering is the phenomenon that the light enters an object at some location, gets scattered and leaves at another location, and hence can be described using the bidirectional surface scattering reflectance distribution function (BSSRDF). BSSRDF shows the relationship between the outgoing radiance, $L_o(x_o, \vec{\omega}_o)$ at the point x_o in direction $\vec{\omega}_o$, and the incident flux,$\Phi_i(x_i, \vec{\omega}_i)$ at the point x_i from direction $\vec{\omega}_i$ $dL_o(x_o, \vec{\omega}_o) = S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) d\Phi_i(x_i, \vec{\omega}_i)$. The well-known bidirectional reflectance distribution function (BRDF) can actually be considered as a simplification of the BSSRDF for which it is assumed that light enters and leaves at the same point (i.e., $x_o = x_i$). The outgoing radiance is computed by integrating the incident radiance over all incoming directions and the whole area A, using a given BSSRDF:

$$L_o(x_o, \vec{\omega}_o) = \int_A \int_{2\pi} S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) L_i(x_i, \vec{\omega}_i)(\vec{n}\vec{\omega}_i) d\vec{\omega}_i dA(x_i)$$

The light transport can be described by radiative transport equation, or called the volume rendering equation:

$$(\vec{\omega}\vec{\nabla})L(x, \vec{\omega}) = -\sigma_t L(x, \vec{\omega}) + \sigma_s \int_{4\pi} p(\vec{\omega}, \vec{\omega}') L(x, \vec{\omega}') d\vec{\omega}' + s(x, \vec{\omega})$$

Here, $s(x, \vec{\omega})$ is the volume source term. Here the medium properties are described by absorption coefficient s_a, absorption coefficient s_s, and the phase function $p(\vec{\omega}, \vec{\omega}')$. The extinction coefficient st is defined as $s_t = s_a + s_s$. The phase function tells the possibility that the light is scattered into a direction $\vec{\omega}$ from the direction $\vec{\omega}'$. We assume that the phase function is normalized ($\int_{4\pi} p(\vec{\omega}, \vec{\omega}') = 1$) and it is only a function of the angle:$p(\vec{\omega}, \vec{\omega}') = p(\vec{\omega}\vec{\omega}')$. The mean cosine, g, of the scattering angle is:

$$g = \int_{4\pi} (\vec{\omega}\vec{\omega}') p(\vec{\omega}\vec{\omega}') d\vec{\omega}'$$

We can change the scattering properties of the medium without significantly influencing the actual distribution of the light. Specifically, we can obtain an isotropic scattering medium (g=0) by changing the scattering coefficient to: $\sigma_s' = (1-g)\sigma_s$ where σ_s' is the reduced scattering coefficient. The absorption coefficient remains unchanged, and we get the reduced extinction coefficient st'=sa+ss'. We can then avoid take into account of the phase functions later.

For an infinitesimal beam entering a homogeneous medium, the incoming radiance will decrease exponentially with the distance s:

$$L_{ri}(x_i + s\vec{\omega}_i, \vec{\omega}_i) = e^{-\sigma_t' s} L_i(x_i, \vec{\omega}_i)$$

The average distance at which the light is extinct the mean-free path, is $l_u = 1/st'$. The diffuse approximation can be found in [Jensen et al. 2002], and just show the diffusion equation obtained from applying the diffusion approximation to the light transport equation:

$$\frac{1}{3\sigma_t'}\nabla^2 F_t(x) = \sigma_a F_t(x) - S_o(x) + \frac{1}{\sigma_t'}\nabla \cdot \vec{S}_1(x)$$

Where the 0^{th}-order spherical harmonic F_t, called the radiant fluence, is $F_t(x) = \int_{4\pi} L(x, \vec{\omega}) d\vec{\omega}$ and the 0^{th}- and the 1^{st}-order spherical harmonics expansion of the source term $s(x, \vec{\omega})$, S_0 and S_1, is $S_o(x) = \int_{4\pi} s(x, \vec{\omega}) d\omega$ and $\vec{S}_1(x) = \int_{4\pi} s(x, \vec{\omega})\vec{\omega} d\omega$, respectively. The diffusion equation above does not have an analytical solution for the general case of finite media. Jensen et al. [2001] uses the dipole diffusion approximation for a point source in a semi-infinite medium. The point source is an approximation of an incoming beam of light for which it is assumed that all light scatters at a depth of one mean-free path below the surface. The dipole diffuse approximation results in the following expression for the diffuse reflectance, the ratio between the radiant exitance M_o at the surface location x_o and the incident flux $F_i(x_i)$ at x_i: $R_d(r) =$

$$\frac{dM_o(x_o)}{d\Phi_i(x_i)} = \frac{\alpha'}{4\pi}\left[z_r(\sigma_{tr}d_r + 1)\frac{e^{-\sigma_{tr}d_r}}{\sigma_t' d_r^3} + z_v(\sigma_{tr}d_v + 1)\frac{e^{-\sigma_{tr}d_v}}{\sigma_t' d_v^3}\right]$$

where, $a' = ss'/st'$ is the reduced albedo, $\sigma_{tr} = \sqrt{3\sigma_a\sigma_t'}$ is the effective transport extinction coefficient, $d_r = \sqrt{r^2 + z_r^2}$ is the distance to the real light source, $d_v = \sqrt{r^2 + z_v^2}$ is the distance to the virtual source, $r = x_0 - x_i$ is the distance from x_o to the point of illumination x_i, and $zr = lu = 1/st'$ and $zv = lu(1 + 4/3A)$ are the distance from the dipole lights to the surface. The A term, counting for the boundary condition for the mismatched interface, is computed as: $A = (1 + Fdr)/(1 - Fdr)$, where the diffuse Fresnel term, Fdr is approximated from the relative index of refraction η: $F_{dr} = -\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta$. After we get the diffuse reflectance, we need to take into account the Fresnel reflection at the boundary for both the incoming light and the outgoing radiance:

$$S_d(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) = \frac{1}{\pi} F_t(\eta, \vec{\omega}_i) R_d(x_i - x_o) F_t(\eta, \vec{\omega}_o)$$

Where Sd is the diffusion term of the BSSDRF. This term only represents multiple scattering. Since the one scattering event is already included in the conversion to the point source approximation, now we need to further recover the single scattering term. The total outgoing radiance, $L_o^{(1)}$, due to single scattering is computed by integrating the incident radiance along the refracted outgoing ray:

$$L_o^{(1)}(x_o, \vec{\omega}_o) = \sigma_s(x_o) \int_{2\pi} F\, p(\vec{\omega}_i \vec{\omega}_o) \int_0^\infty e^{-\sigma_{tr} s} L_i(x_i, \vec{\omega}_i) ds d\omega_i$$

$$= \int_A \int_{2\pi} S^{(1)}(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) L_i(x_i, \vec{\omega}_i)(\vec{n}\vec{\omega}_i) d\omega_i dA(x_i)$$

Here $F = F_t(\eta, \vec{\omega}_o) F_t(\eta, \vec{\omega}_i)$ is the product of the two Fresnel transmission terms, and $\vec{\omega}_i$ and $\vec{\omega}_o$ are the refracted incoming and outgoing directions. The combined extinction coefficient stc is given by $stc = st(xo) + Gstc(x_i)$, where G is a geometry factor; for a flat surface $G = \frac{|\vec{n}_i \vec{\omega}_o|}{|\vec{n}_i \vec{\omega}_i|}$. The single scattering BSSRDF, $S(1)$, is defined implicitly by the second line of this equation. Note that there is a change of variables between the first line, which integrates only over the configurations where the two refracted rays intersect, and the second line, which integrates over all incoming and outgoing rays. This means the distribution S(1) contains a delta function.

The complete BSSRDF model is the sum of the diffusion approximation and the single scattering term:

$$S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) = S_d(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) + S^{(1)}(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o)$$

This model accounts for light transport between different locations on the surface, and it has both the direction component (due to the single scattering) as well as the diffuse component (due to the multiple scattering). Unfortunately, this is rather slow for a complex scene with fracture. We use a russian roulette acceleration structure to push the implementation to realtime.

2.2 Russian Roulette Acceleration Structure

We use the discrete Monte Carlo estimator for rays with low expected values, instead of just abandoning them, to make sure that the estimated value is unbiased. We can afford tracing all rays with recursive depth less than some threshold level dt. Then for rays deeper than that threshold, only choose one path for each light transport event: given one uniform random value ξ in $[0, 1]$, if

$$\xi < f(\omega_o, \omega_i)/(f(\omega_o, \omega_i) + f(\omega_r, \omega_i))$$

, we choose the reflection direction only, and the reflection intensity will be weighted by $(f(\omega_o, \omega_i) + f(\omega_r, \omega_i))/f(\omega_o, \omega_i)$. Otherwise, we only trace the refraction direction and weight the intensity by $(f(\omega_o, \omega_i) + f(\omega_r, \omega_i))/f(\omega_r, \omega_i)$. Here f is the BSSRDF. ω_i is the original direction, ω_o is the reflected direction and ω_r is the refracted direction. The estimator is unbiased, since:

$$E\left[\frac{f(\vec{\omega}_r, \vec{\omega}_i)}{f(\vec{\omega}_o, \vec{\omega}_i) + f(\vec{\omega}_r, \vec{\omega}_i)} \frac{f(\vec{\omega}_o, \vec{\omega}_i) + f(\vec{\omega}_r, \vec{\omega}_i)}{f(\vec{\omega}_r, \vec{\omega}_i)} L_i(\vec{\omega}_r, \vec{\omega}_i) + \frac{f(\vec{\omega}_o, \vec{\omega}_i)}{f(\vec{\omega}_o, \vec{\omega}_i) + f(\vec{\omega}_r, \vec{\omega}_i)} \frac{f(\vec{\omega}_o, \vec{\omega}_i) + f(\vec{\omega}_r, \vec{\omega}_i)}{f(\vec{\omega}_o, \vec{\omega}_i)} L_i(\vec{\omega}_o, \vec{\omega}_i)\right] = L_i(\vec{\omega}_r, \vec{\omega}_i) + L_i(\vec{\omega}_o, \vec{\omega}_i)$$

If the ray goes deeper than some maximum depth d_{max}, we can use Russian roulette technique to enforce tracing termination. The method is similar to the Monte Carlo method with importance sampling, except that the domain is discrete: only two directions, specular reflection and refraction; and only one direction need to decide to trace. In survey sampling literature, the discrete estimator is known as Horvitz estimator[Horvitz and Thompson 1952]. When the tracing depth grows deeper, we assume that the light intensity distribution is decreasing and diffused after many reflection/refraction events. The BSSRDF function can be used to approximate the real light intensities for importance sampling. If dmax is large enough that most rays can get terminated before reaching the maximum depth, it is reasonable to abandon those deep rays since it will not greatly affect the final estimates. But Russian roulette technique can be still used to guarantee an unbiased estimator. Randomly choose a number ξ between $(0, 1)$ and compare it with some termination value P_s. If $\xi < P_s$, terminate it immediately, otherwise, continue tracing the ray and weight it by $\frac{1}{(1-Ps)}$. It is unbiased as proved before. Increasing the threshold depth dt can help reduce the noises caused by the random samplings, but on the other hand, the total ray number will increase approximately by a factor 2.

3 Conclusion

We proposed a novel framework for fast and robust simulation and rendering of brittle fracture with extreme complexity.

Figure 4: Large Fracture Simulation of the Explosion of the Buddha's head (500K tetrahedra).

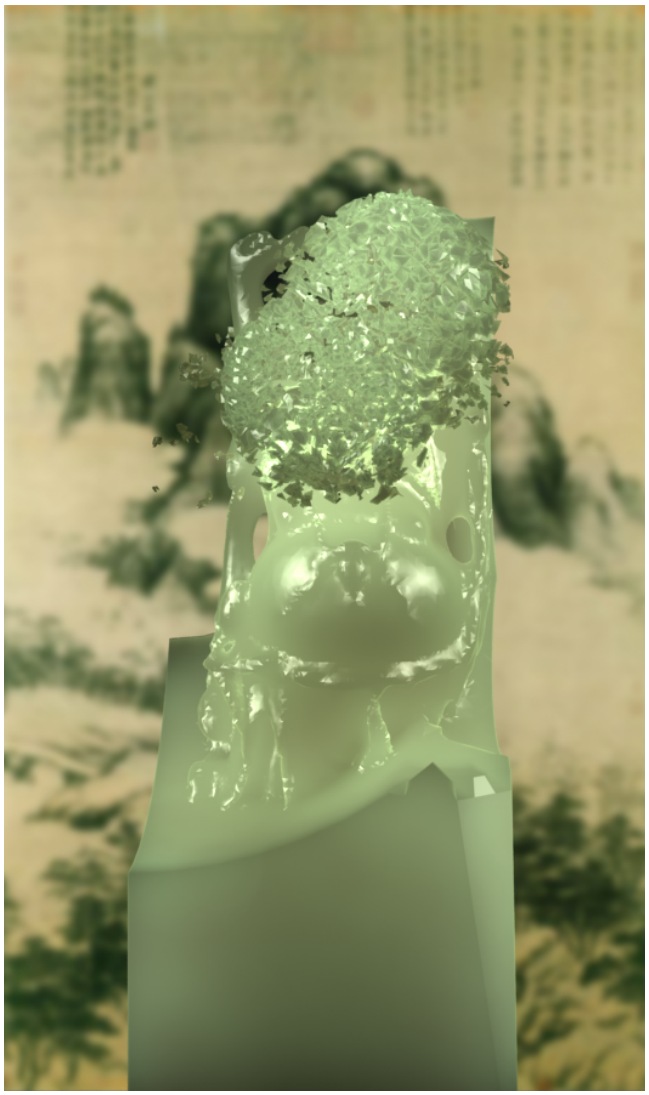

Figure 5: Large Fracture Simulation of the Explosion of the Buddha's head (500K tetrahedra).

References

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 586–593.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A multiresolution framework for dynamic deformations. In *ACM SIGGRAPH Symp. on Comput. Anim.*, ACM Press, 41–48.

CHEN, D., AND ZELTZER, D. 1992. Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method. *Comput. Graph. (SIGGRAPH Proc.)*, 89–98.

DEBUNNE, G., DESBRUN, M., CANI, M., AND BARR, A. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proc. SIGGRAPH 2001*, vol. 20, 31–36.

GOURRET, J.-P., MAGNENAT-THALMANN, N., AND THALMANN, D. 1989. Simulation of object and human skin deformations in a grasping task. *Comput. Graph. (SIGGRAPH Proc.)*, 21–30.

HIROTA, K., TANOUE, Y., AND KANEKO, T. 1998. Generation of crack patterns with a physical model. *The Vis. Comput. 14*, 126–187.

HORVITZ, D., AND THOMPSON, D. 1952. A generalization of sampling without replacement from a finite universe. 47: 663–685.

JENSEN, H. W., MARSCHNER, S., LEVOY, M., AND HANRAHAN, P. 2002. A Practical Model for Subsurface Light Transport. In *Proc. of SIGGRAPH 2002*, 511–518.

MAZARAK, O., MARTINS, C., AND AMANATIDES, J. 1999. Animating exploding objects. In *Proc. of Graph. Interface 1999*, 211–218.

MOLINO, N., BAO, J., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. (SIGGRAPH Proc.), (in press)*.

MULLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Grap. Interface*.

MULLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Comput. Anim. and Sim. '01*, Proc. Eurographics Workshop, Eurographics Assoc., 99–111.

MULLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *ACM SIGGRAPH Symp. on Comput. Anim.*, 49–54.

MULLER, M., TESCHNER, M., AND GROSS, M. 2004. Physically-based simulation of objects represented by surface meshes. In *Proc. Comput. Graph. Int.*, 156–165.

Figure 6: Large Fracture Simulation of the Explosion of the Buddha's head (500K tetrahedra).

Figure 7: Large Fracture Simulation of the Explosion of the Buddha's head (500K tetrahedra).

NEFF, M., AND FIUME, E. 1999. A visual model for blast waves and fracture. In *Proc. of Graph. Interface 1999*, 193–202.

NORTON, A., TURK, G., BACON, B., GERTH, J., AND SWEENEY, P. 1991. Animation of fracture by physical modeling. *Vis. Comput. 7*, 4, 210–219.

O'BRIEN, J., AND HODGINS, J. 1999. Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH 99*, vol. 18, 137–146.

O'BRIEN, J., BARGTEIL, A., AND HODGINS, J. 2002. Graphical modeling of ductile fracture. *ACM Trans. Graph. (SIGGRAPH Proc.) 21*, 291–294.

SMITH, J., WITKIN, A., AND BARAFF, D. 2001. Fast and controllable simulation of the shattering of brittle objects. In *Comput. Graphics Forum*, D. Duke and R. Scopigno, Eds., vol. 20(2). Blackwell Publishing, 81–91.

TERAN, J., SALINAS-BLEMKER, S., NG, V., AND FEDKIW, R. 2003. Finite volume methods for the simulation of skeletal muscle. In *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 68–74.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Comput. Graph. (Proc. SIGGRAPH 87) 21*, 4, 205–214.

YNGVE, G., O'BRIEN, J., AND HODGINS, J. 2000. Animating explosions. In *Proc. SIGGRAPH 2000*, vol. 19, 29–36.