6
Physics-based Fracture and Fragmentation Simulation

Our method for cohesive fracture and fragmentation simulation is computationally expensive from the numerical point of view. Moreover, the time step used is extremely small to avoid instability (below nano seconds). Therefore our simulation would not be acceptable in a physics-based animation because it would take a long time to simulate the rigid body dynamics.

Recent works addressed the use of fracture animation in Computer Graphics with finite element meshes. In 2013, Busarye et at. [25] perform a realistic animation of thin plate fracture by relaxing stress to handle multiple fracture cuts in a single step and to prevent objects from fracturing into small pieces. Their mesh surface is discretized as triangular elements rather than three-dimensional elements (our case). They use an adaptive fracture-aware remeshing scheme based on constrained Delaunay triangulation to produce fracture details. Like our two-dimensional adaptive fracture simulation (see chapter 4), a local refinement is performed where the fracture is about to be initiated. They refine for unstructured meshes while we use specialized 4K mesh refinement and coarsening. Like [25], Pfaff et al. [27] also use triangular meshes to propose an adaptive method propagation in thin sheets, as well as refine the mesh where cracks are likely to start or advance. Refinement allows an efficient simulation by reducing the total number of nodes and elements of the mesh, but coarsening is also done to reduce even more that number of entities. Both works deal with refinement, which ensures that the stress distribution around the crack tip is well resolved. Pfaff et al. include bending and stretching plasticity models to simulate a large range of materials with different fracture behaviors.

In three-dimensional simulations, Koschier et al. [28] use a novel reversible tetrahedral mesh refinement scheme for adaptive simulation of brittle fracture of solid objects. Because most brittle materials slightly deform before the crack originates, they model the dynamic behavior using a rigid body dynamics simulator. High internal stresses are consequences of contact with other objects. During the contact with the objects, regions with high stress are refined. Fracture occurs if the tensile stress exceeds a certain threshold and mesh parts are separated by a fracture surface represented by a signed distance function. If the stress regions relax, they coarsen the region's discretization. Using a different method, Chen et al. [29] adaptively refine and coarsen a fracture surface into a detailed one, based on a discrete gradient descent flow. They are based on a physics-inspired approach to enrich low-resolution fracture animation by realistic fracture details. They use a implicit corotational FEM method by Müller et al. [76] to simulate the dynamics and use the method proposed by O'Brien and Hodgins [77] to generate fracture cuts. After the extraction, the low resolution fracture surface iteritevely evolves in the material space using a gradient flow by deforming and adaptively remeshing it.

Our proposed method also splits the simulation into rigid body dynamics and handles fracture simulation. In this chapter, we propose the incorporation of an existent method to increase the time step of our simulation and handle rigid body dynamics in animation of three-dimensional quasi-brittle and brittle objects using our traditional CZM simulation. Much like previous works, we handle rigid body dynamics and fracture simulation separately. But none these works to our knowledge use the Cohesive Zone Model.

6.1 Our approach

Our mesh is composed of tetrahedra (Tet4) elements and simulated using the Cohesive Zone Model. Our approach consists of dividing the simulation in two parts: *rigid body mode* and *fracture mode*. In the rigid body mode, we handle the rigid body dynamics and do not fracture the object(s), i.e., only deal with global and individual nodal motion such as linear and angular momentum and collision detection. The fracture mode handles the fracture and fragmentation simulation whenever a collision is detected. We simulate this mode with a lower time step than the rigid body mode, but increased time step compared to our simulations from previous chapters (refer to chapters 3, 4, and 5). Our simulation for both modes is based on the CZM simulations implemented in chapters 3, 4, and 5. In the next sections, we will show how the time step can be increased for both modes and at the same time avoid instability. Our animation framework is limited to a non real-time simulation since we need to adapt the time steps between the two modes.

Consider the example of dropping an arbitrary mesh with only the force of gravity acting on it, like the sphere in Figure 6.1. At this moment, since no collision is detected, the rigid body dynamics simulation acts on the model. Velocities and accelerations are calculated just like the CZM using the explicit integrator scheme (refer to equations 2.1 and 2.2), but no stress or internal force is computed and used as contribution to nodal accelerations. When the first nodes of the mesh collide with the obstacles, for example, a static plane (see Figure 6.3), we switch to fracture mode. The nodes that collide with the obstacles are "pulled" back in the direction of the plane´s normal. This

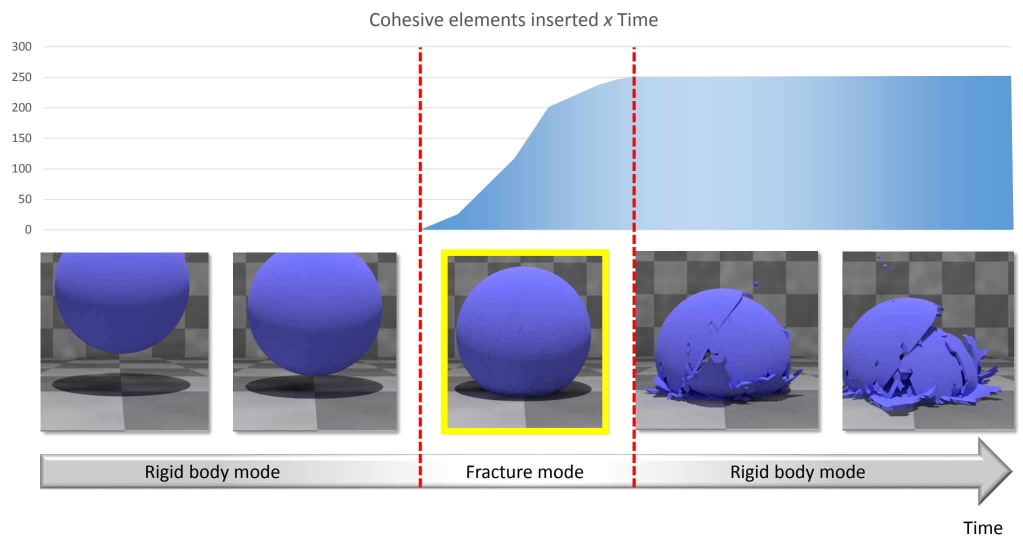


Figure 6.1 – The figure illustrates our framework to deal with physics-based animation of brittle and quasi-brittle objects. The objects starts falling in rigid body mode. When it collides, it switches to fracture mode and cohesive elements are inserted. When no cohesive elements are added in a while, it switches back to rigid body mode.

displacement causes stress at the nodes that can generate possible fractured facets and insertion of cohesive elements (i.e. fracture begins propagating). The stress calculation and insertion of cohesive elements verification is an effect of the fracture mode that are not present in the rigid body mode. We switch back to the rigid body mode when no cohesive element is added to the mesh at a certain number of time steps. Notice that we are still using the CZM even when in rigid body mode when calculating displacements, velocities, and accelerations using our typical explicit integration scheme. We leave an automatic switching between both modes based on the strain energy as future work.

6.2 Fracture mode and constraint dynamics

In the previous chapters (refer to chapters 3, 4, and 5), the time steps chosen for the fracture and fragmentation simulations were based on the Courant-Friedrichs-Lewy (CFL) condition. They were in range of nano seconds, which is not reasonable for a physics-based simulation. Our proposal is to increase the simulation time step and at the same time avoid instability that would happpen when violating the CFL condition.

We use Müller et al.'s position based dynamics method [2] to created a rigid body connecting all nodes of our mesh by a constrained rigid bar. If the nodes try to separate too much from each other, a phenomenum caused by instability in the CFL condition, we use the rigid bar constraint to “pull” them back towards

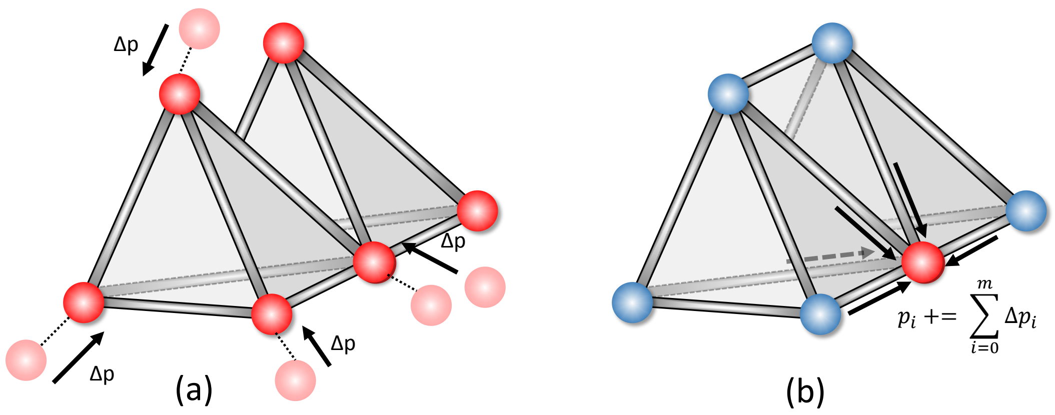


Figure 6.2 – The figure illustrates the nodes belonging to the tetrahedra elements and connected by rigid bars. We use relaxation method with constraint functions proposed by Müller et al. to avoid time step instability. (a) shows each node moving away too much from its resting position as an effect of instability. (b) shows the correction we have to make in one node (m is the number of neighbor bars of the node).

each other in the direction of the bar. This is an iterative/relaxation method, as illustrated by Figure 6.2, since each node is connected to more than one bar. The relaxation loop also includes treating collision response: for example, by "pulling" the node back inside the plane in the direction of its normal with a depth equal to the node-to-plane distance. This displacement determines an increase in stress, which may generate fracture propagation in the mesh, as explained earlier. Figure 6.3 illustrates a physics-based animation of a bunny under gravity action and colliding with the ground. It is treated as a rigid body, except during collision when it fractures. The model is composed by 69,668 nodes and 208,353 bulk elements. Initial material parameters are as follows: initial velocity = 0 m/s, elastic modulus = 0.6 Pa, Poisson coefficient = 0.23, specific mass = 2400 kg/m3. Fracture energy materials are as follows: fracture energy $\mathrm{G_I}$ = 22 N/m, cohesive strength smax = 0.1 mPa, and shape parameter α = 2. Time step is increased to 10e-4 s in rigid body mode and decreased to 5 μs in fracture mode, with stress calculated at every time step. Notice that even with a low time step for the fracture mode, it is much higher than the typical time steps for the CZM simulation.

Müller et al. define general constraints via a constraint funtions [78, 79] to simulate dynamic objects, such as rigid bodies. Instead of computing forces as derivative of a constraint function energy, they solve directly for the equilibrium configuration and project positions [2]. We refer the reader to Müller et al.'s algorithm for further details [2] as we take advantage of their method to increase our simulation time step during fracture mode and to replicate rigid bodies during rigid body mode..

The definition of projecting a set of points according to a constraint

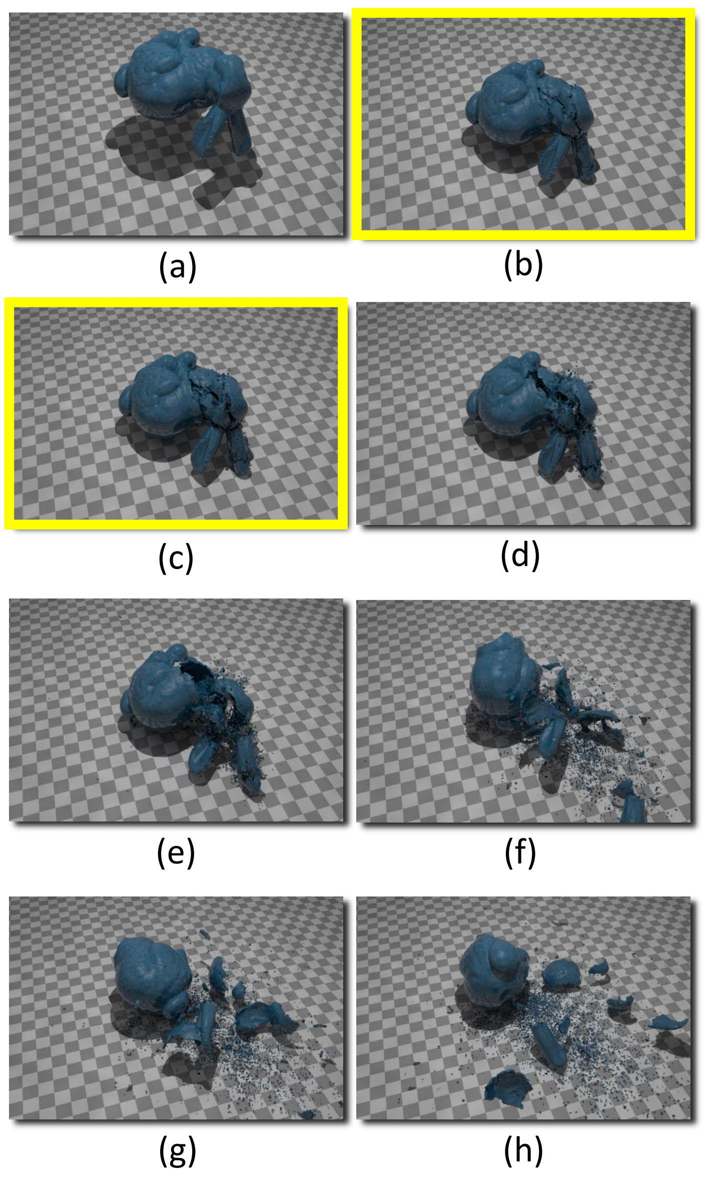


Figure 6.3 – Bunny model with 69,668 nodes and 208,353 bulk elements. Initial material parameters are as follows: initial velocity = 0 m/s, elastic modulus = 0.6 Pa, Poisson coefficient = 0.23, specific mass = 2400 kg/m3. Fracture energy materials are as follows: fracture energy G_I = 22 N/m, cohesive strength smax = 0.1 mPa, and shape parameter $\alpha = 2$. Time step is 10e-4 s, with stress calculated at every time step. Position-based parameters are: $k_{stiffness} = 0.9$, $k_{damping} = 1$ and number of iterations = 3. (a) The bunny begins falling under gravity action in rigid body mode. No fracture propagates in the model, although the simulation is done using the Cohesive Zones Model (CZM) (i.e. using explicit integration) and Müller´s Positions-based Dynamics (to avoid instability with increased time step); (b) The bunny collides with the ground. When the first node(s) collide(s), we switch to fracture mode. Stress calculation lead to cohesive elements verification, which leads to node duplication and fracture propagation; (c, d) The bunny continues in fracture mode because cohesive elements are still being added to the mesh; (e, f, g, h) Cohesive elements were not added at a certain number of steps, so the simulation is switched to rigid body mode. No fracture will propagate in the mode. Instead, each separate component of the bunny will be treated as a rigid body.

function is to move them so they satisfy the constraint, by conserving both linear and angular momentum. Let $\Delta\mathbf{p}_i$ be the displacement of node i by the projection. Linear momentum is conserved if

$$\sum_i m_i \Delta\mathbf{p}_i = 0 \tag{6.1}$$

Angular momentum is conserved if

$$\sum_i \mathbf{r}_i \times m_i \Delta\mathbf{p}_i = 0 \tag{6.2}$$

where $\mathbf{r}_i$ are the distances of $\mathbf{p}_i$ to the center of mass of the object.

The method proposed by [2] conserves both linear and angular momenta for internal constraints. Let C be the constraint with stiffness k and cardinality n on points $\mathbf{p}_1, ..., \mathbf{p}_n$. The stiffness k defines the strength of the constraint and ranges from zero to one, with one being the most rigid. Given $\mathbf{p}$, we want to find the correction $\Delta\mathbf{p}$ such that $C(\mathbf{p} + \Delta\mathbf{p}) = 0$, which can be approximated by

$$C(\mathbf{p} + \Delta\mathbf{p}) \approx C(\mathbf{p}) + \nabla_{\mathbf{p}} C(\mathbf{p}) \cdot \Delta\mathbf{p} = 0 \tag{6.3}$$

By choosing a scalar λ to restrict $\Delta\mathbf{p}$ to be in the direction of $\nabla_{\mathbf{p}} C(\mathbf{p})$, we have

$$\Delta\mathbf{p} = \lambda \nabla_{\mathbf{p}} C(\mathbf{p}) \tag{6.4}$$

Sybstituting Eq. 6.4 into Eq. 6.3, solving for λ and substituting back into Eq. 6.4, we get the final formula for $\Delta\mathbf{p}$

$$\Delta\mathbf{p} = -\frac{C(\mathbf{p})}{|\nabla_{\mathbf{p}} C(\mathbf{p})|^2} \nabla_{\mathbf{p}} C(\mathbf{p}) \tag{6.5}$$

For the correction of an individual point $\mathbf{p}_i$ we have

$$\Delta\mathbf{p} = -s \nabla_{\mathbf{p}} C(\mathbf{p}_1, ..., \mathbf{p}_n) \tag{6.6}$$

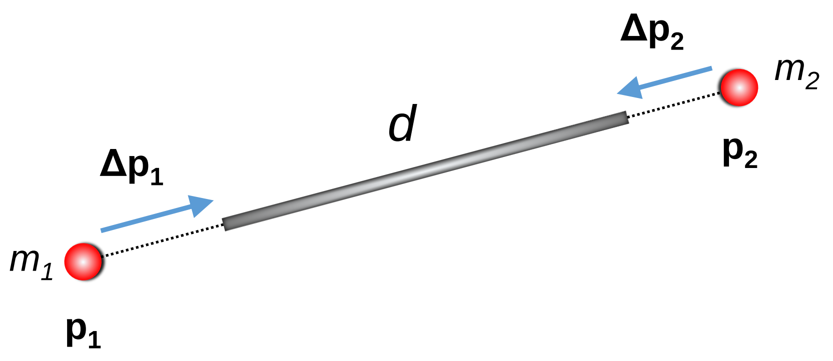


Figure 6.4 – Rigid bar constraint and the displacements needed to "pull back" the nodes to the rest distance position. The projection of the constraint $C(p_1, p_2) = |p_1 - p_2| - d$, where the corrections $\Delta\mathbf{p}_i$ are weighted by the inverse masses $1/m_i$.

where the scaling factor

$$s = \frac{C(\mathbf{p}_1, ..., \mathbf{p}_n)}{\sum_j \left|\nabla_{\mathbf{p}_j} C(\mathbf{p}_1, ..., \mathbf{p}_n)\right|^2} \tag{6.7}$$

is the same for all nodes. For nodes of individual masses, we weight the corrections $\Delta\mathbf{p}_i$ by the inverse masses $w_i = 1/m_i$. Equations 6.8 and 6.8 are replaced by

$$\Delta\mathbf{p} = -s w_i \nabla_{\mathbf{p}} C(\mathbf{p}_1, ..., \mathbf{p}_n) \tag{6.8}$$

$$s = \frac{C(\mathbf{p}_1, ..., \mathbf{p}_n)}{\sum_j w_i \left|\nabla_{\mathbf{p}_j} C(\mathbf{p}_1, ..., \mathbf{p}_n)\right|^2} \tag{6.9}$$

The constraint is equivalent to pulling the nodes with a "ghost" force. As an example which we use in our simulations, our constraint is defined as a rigid bar connecting each neighbor node of the mesh. Figure 6.4 illustrates the rigid bar constraint and the displacements needed to "pull" back the nodes to the rest distance. The rigid bar constraint can be described as $C(p_1, p_2) = |p_1 - p_2| - d$. The derivative with respect to the nodes are $\nabla_{\mathbf{p}_1} C(\mathbf{p}_1, \mathbf{p}_2) = \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|}$ and $\nabla_{\mathbf{p}_2} C(\mathbf{p}_1, \mathbf{p}_2) = -\frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|}$. The scaling factor s is $s = \frac{|\mathbf{p}_1 - \mathbf{p}_2| - d}{w_1 + w_2}$. The final corrections are:

$$\Delta\mathbf{p}_1 = -\frac{w_1}{w_1 + w_2}(|\mathbf{p}_1 - \mathbf{p}_2| - d)\frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \tag{6.10}$$

$$\Delta\mathbf{p}_2 = +\frac{w_2}{w_1 + w_2}(|\mathbf{p}_1 - \mathbf{p}_2| - d)\frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \tag{6.11}$$

The stiffness constant ($k_{stiffness}$) multiplies $\Delta\mathbf{p}$ between 0 and 1, which determines the amount of stiffness of the mesh. We do not want any instability in the model, but we also do not want to lose the material properties imposed by elasticity, so having the stiffness constant set to a factor lower than 1 is a reasonable idea.

6.3 Rigid body simulation

When the object is colliding, we use the traditional CZM simulation framework. When treating the object as a rigid body (i.e. when not colliding), we do not calulate stresses nor duplicate nodes, neither use internal forces contributions in the acceleration in equation 2.1. But still, the object might deform because it may lack rigid bar connections between certain nodes. For example, as hollow sphere might deform as its northern hemisphere is not connected to its southern hemisphere. To avoid this behavior and treat objects as rigid, we use Müller et al's global damping technique [2] without influencing rigid body modes of the object. They compute the global linear velocity and angular velocity of each component/system and damp only the individual deviations of the velocities from the global motion. The amount of damping is controled via an user-defined constant called $k_{damping}$ that ranges from zero to one. Thus, with $k_{damping} = 1$, only global motion survives and the set of vertices of each component behaves like a rigid body. We label the connected components of the mesh in order to calculate each center of mass $\mathbf{x}_{cm}$ and then calulate the velocity change for each node in each component. The equations for damping are listed in table 6.1.

```
1: $\mathbf{x}_{cm} = (\sum_i \mathbf{x}_i m_i)/(\sum_i m_i)$
2: $\mathbf{v}_{cm} = (\sum_i \mathbf{v}_i m_i)/(\sum_i m_i)$
3: $\mathbf{L} = \sum_i \mathbf{r}_i \times (m_i \mathbf{v}_i)$
4: $\mathbf{I} = -\sum_i \tilde{\mathbf{r}}_i \tilde{\mathbf{r}}_i^T m_i$
5: $\omega = \mathbf{I}^{-1}\mathbf{L}$
6: for all vertices $i$ do
7:     $\Delta\mathbf{v}_i = \mathbf{v}_{cm} + \omega \times \mathbf{r}_i - \mathbf{v}_i$
8:     $\mathbf{v}_i \leftarrow \mathbf{v}_i + k_{damping}\Delta\mathbf{v}_i$
9: end for
```

Table 6.1 – Damping calculation for rigid bodies [2].

Line 1 calculates the center of mass $\mathbf{x}_{cm}$ of the object/component, where $\mathbf{r}_i = \mathbf{x}_i - \mathbf{x}_{cm}$. Line 2 calculates the velocity of the center of mass of the object/component. Line 3 computes the angular momentum $\mathbf{L}$. Line 4 computes the moment of inertia $\mathbf{I}$, where $\tilde{\mathbf{r}}_i$ is the 3×3 matrix with property $\tilde{\mathbf{r}}_i\mathbf{v} = \mathbf{r}_i \times \mathbf{v}$. Using the inverse $\mathbf{I}^{-1}$ and $\mathbf{L}$, we compute the angular velocity ω at line 5. Lines 6 through 9 damp the individual deviations $\Delta\mathbf{v}_i$ of the velocities $\mathbf{v}_i$ from the global motion $\mathbf{v}_{cm} + \omega \times \mathbf{r}_i$. Therefore, with $k_{damping} = 1$, the component behaves as a rigid body.

6.4 Collision detection and response

The collision detection and response is simply another constraint added to our list of projected positions inside the relaxation loop and is based on checking object-to-obstacle interpenetration. If the object collided, we must deal with the collision response. In this case, we are verifying if each node of the mesh collided to an obstacle. To check the node-to-object collision, we calculate the distance between the node and the obstacle in the direction of the interpenetration vector. If the node has penetrated the obstacle, we signal a collision response and switch to fracture mode. At this point, the collision response will be calculated inside the relaxation loop of the projected constraints. We simply “pull” the node back in the direction of the obstacle’s normal at the interpenetration point with a distance of the depth of the interpenetration. In the next iteration, we repeat the procedure since the relaxation may have moved the nodes inside the obstacle again.

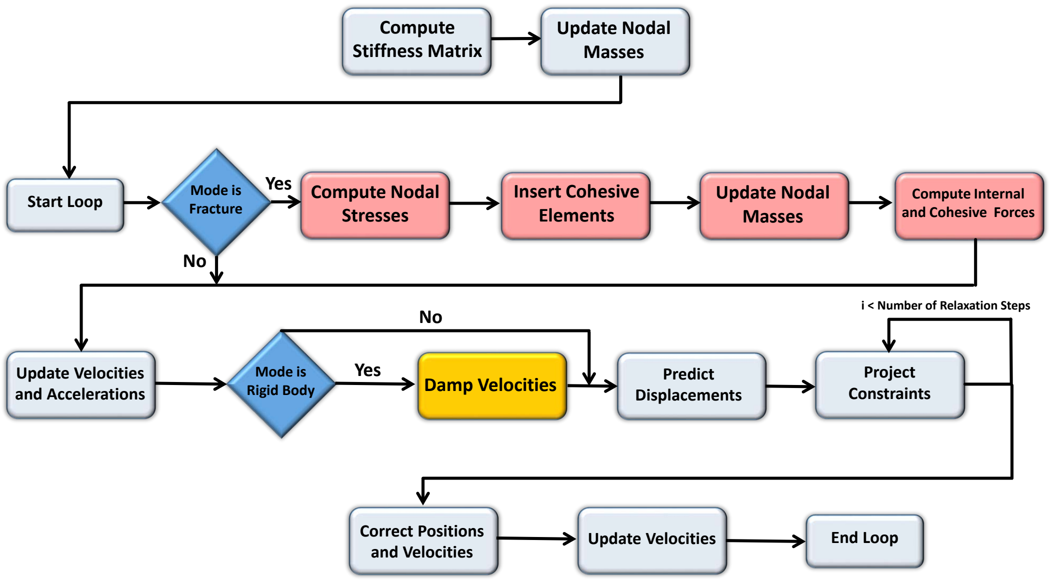


Figure 6.5 – The figure illustrates the simulation flow chart. According to the chart, we can see clearly when the fracture and rigid body modes act on the simulation, which uses the same procedure from the CZM simulation, combined with Müller et al.'s constraint projection to avoid instability.

6.5 Physics-based simulation

Figure 6.5 illustrates a flow chart of the simulation. We start pre-processing the stiffness matrices and the nodal masses and enter the simulation loop. From here on, we switch between fracture and rigid body modes depending on collision detection and cohesive elements insertion. Our framework uses the same procedure from the CZM simulation, combined with Müller et al.'s constraint projection to avoid instability.

The position based dynamics method incorporated to our framework is extremely relevant because it allows us to increase the time step of our fracture mode simulation while maintaining numerical stability. If the nodes try to move away too much from each other, we "pull" them back in the direction of the incident bars using the relaxation method for m neighbor bars. Because we do not represent edges explicitely in our data structure, our algorithm is based per element (tetrahedron) and on coloring. We launch a kernel with a thread per element using the mesh coloring to avoid race condition. Each thread processes all of its tetrahedron edges and accumulates $k_{stiffness} \cdot \Delta\mathbf{p}$ on its rigid bars' (edges) nodal positions.

Table 6.2 shows the algorithm combining our fracture and fragmentation technique with Müller's position based dynamics method. Line 1 initializes the stiffness matrix and the lump mass matrix and line 2 updates the nodal masses exactly like the Cohesive Zone Model (CZM). Lines 4 and 5 initialize all nodal positions, velocities and weights used by the constraints. The weights refer to the inverse masses $1/m_i$. Line 7 begins the simulation loop. At line 8, if fracture mode is detected (i.e. the object collided to an obstacle), stresses are calculated at line 10, cohesive elements are checked for insertion and node duplications are done if needed at line 12. Nodal masses are then update at line 13. Internal forces are computed at line 16. At line 18, we update velocities and accelerarations like the CZM using an explicit integrator scheme (refer to section 2.2). Until this point, we have described all simulations steps exactly like the CZM, with the exception of updating displacements. Line 19 checks if we are in rigid body mode (i.e. no fracture is propagating). If so, we use Müller et al.´s equations in table 6.1 to damp individual deviations of nodal velocities and treat the object as a rigid body. From line 22 forward, all steps are executed independent if we are in fracture or rigid body modes. At line 22, we predict the displacements using the explicit integrator scheme using our CZM integrator. Lines 23 to 25 refer to the relaxation steps required to reach stability and "pull back" the nodes to their rest distance. During a number of steps, we pull them back with $k_{stiffness} \cdot \Delta\mathbf{p}$ (Equations 6.10, 6.11). We also include collision corrections by pulling nodes back in the direction of the obstacle's normal. Lines 26 to 29 update, for each node, the new velocity and positions using the projected positions $\mathbf{p}_i + \Delta\mathbf{p}_i$ and the previous positions $\mathbf{x}_i$. Finally, at line 30, we update the velocities according to friction restitution.

```
1:  Compute Stiffness Matrix
2:  Update Nodal Mass
3:  current step $\leftarrow 0$
4:  for all vertices $i$ do
5:      Initialize $\mathbf{x}_i, \mathbf{v}_i, \mathbf{w}_i$
6:  end for
7:  while current step $<=$ maximum step do
8:      if mode $==$ FRACTURE then
9:          if current step $==$ check step then
10:             Compute Stresses
11:             if stresses $>$ stress threshold then
12:                 Insert Cohesive Elements
13:                 Update Nodal Masses
14:             end if
15:         end if
16:         Compute Internal Forces
17:     end if
18:     Update Velocities and Accelerations
19:     if mode $==$ RIGID BODY then
20:         Damp Velocities
21:     end if
22:     Predict Displacements $\mathbf{p}_i$
23:     while $i <$ number relaxation steps do
24:         Project Constraints$(C_1, ..., C_{collision}, \mathbf{p}_1, ..., \mathbf{p}_N)$
25:     end while
26:     for all vertices $i$ do
27:         $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
28:         $\mathbf{x}_i \leftarrow \mathbf{p}_i$
29:     end for
30:     Update Velocities
31:     current step $=$ current step $+ 1$
32: end while
```

Table 6.2 – Proposed Physics-based simulation algorithm.

6.6
Experimental results

In this section, we propose a verification of the physics model to use our fracture and fragmentation animation of general tetrahedra meshes. Next, we compare different physical parameters used in the simulation that affect cracks and stiffness, and discuss further examples. All images from animation examples were rendered using the Mitsuba physically based renderer[1]. Mitsuba is an open-source, research-oriented rendering system which generates physically-based rendering images from scene description files.

6.6.1
Verification of the physical model

In order to guarantee an animation that uses physics-based techniques and is consistent with physics and engineering equations, we verify the original 3-dimensional mixed-mode beam from previous experiments by including Müller et al's [2] Position-based Dynamics technique using constrained bars and increasing the time step. No damping was applied to the model since we only want to verify the fracture propagation and not treat it as a rigid body.

We have tested the 3D simulation on the coarse version of the mixed-mode beam. The coarse version contains 113,984 bulk elements and 25,777 nodes. We employed Tet4 (tetrahedra) elements on the mesh and T3 cohesive surface elements. The greedy algorithm was used to color the mesh and the number of colors obtained were 55 on the mesh. The mesh is initially refined in the middle left where the fracture tends to propagate. Cohesive elements were checked for insertion at each 10 steps of the simulation. The coarse mesh had a time step of 3e-5 and 300 steps, while the previous version had a 30 ns time step and 9,000 steps. Initial material parameters are: initial velocity = 25 m/s, elastic modulus = 9 MPa, Poisson coefficient = 0.24, specific mass = 2400 kg/m3. Cohesive fracture parameters are as follows: fracture energy $\mathrm{G_I}$ = 22 N/m, cohesive strength smax = 8 kPa, and shape parameter $\alpha = 2$. In terms of physics-based animation parameters, we used four relaxation iterations and $k_{stiffness} = 1$ to guarantee full stability of the model.

Figure 6.6 shows the result obtained. The model indeed stabilized with a time step 1,000 times greater than the previous version. The fracture propagated from the initial notch through a 30° angle. The simulation took 300 steps to run the full fracture propagation.

[1]https://www.mitsuba-renderer.org/

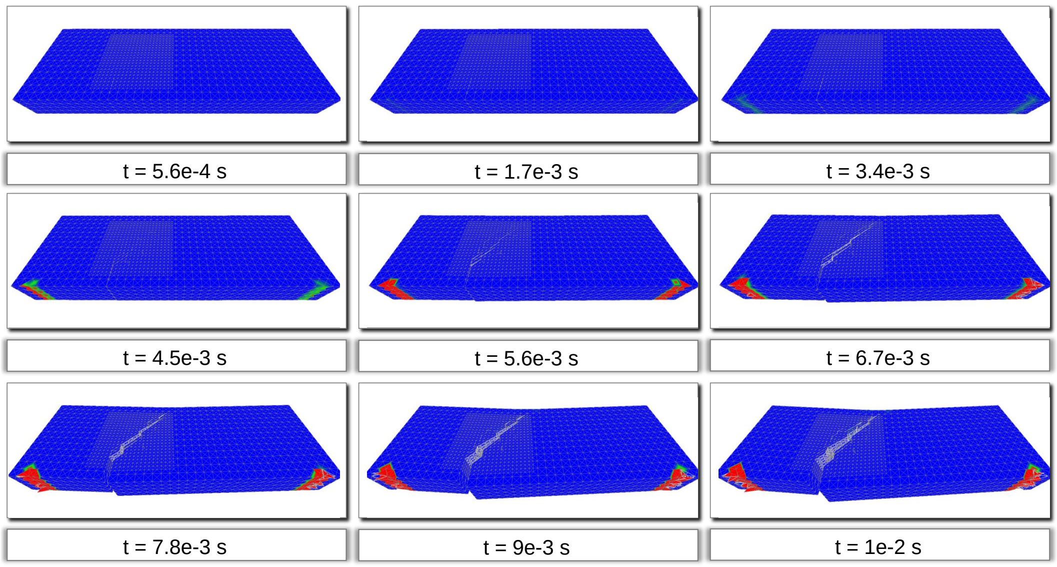


Figure 6.6 – Verification of the engineering model with the relaxation method proposed by Müller et al [2] with time step 1,000 times greater than the versiona from chapter 5.7.

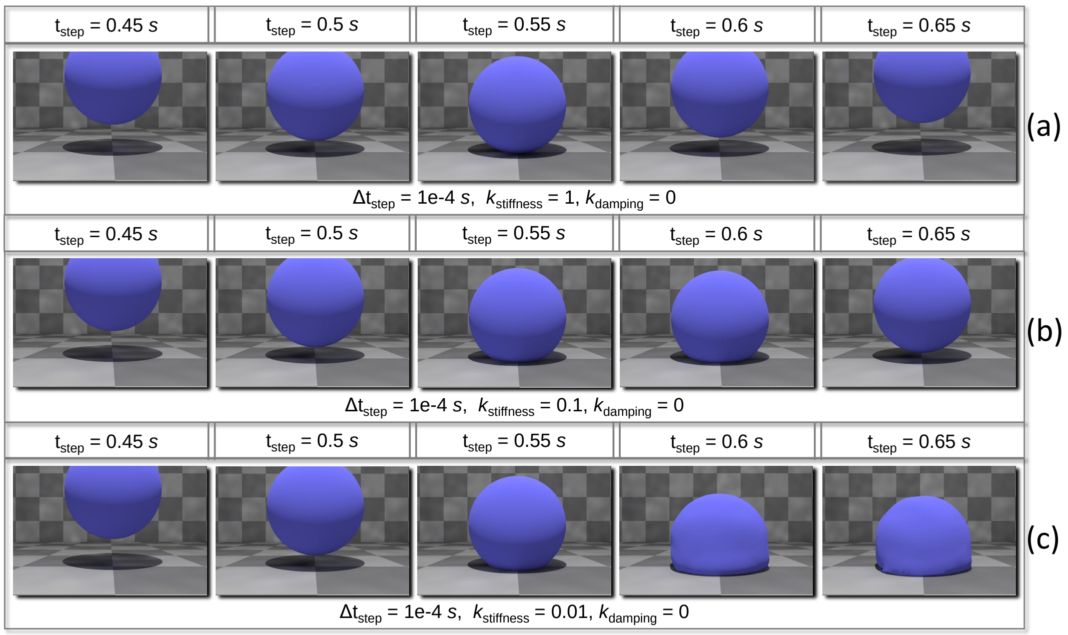
6.6.2 Parameter comparisson

Several parameters can influence the simulation stability and the mesh deformation, such as the size of the time step, the stiffness ($k_{stiffness}$) and damping ($k_{damping}$) coefficients, respectively. We noticed that the greater the time step, the more likely the simulation is in becoming unstable. Moreover, the size of the time step determines the amount of deformation of an object if no damping is applied to velocities. Consider a hollow sphere. We know that each node is connected to its neighbour, but the northern and southern hemispheres are not connected at all by a rigid bar. During collision, the tendency is having the sphere deform if the time step is large enough.

Figure 6.7 shows three different simulations for a hollow tetrahedron sphere falling to the ground with no fracture or cohesive insertion. In simulation (a), the time step is 10e-4 s and $k_{stiffness}$ is 1, or the most rigid system. Notice that the sphere does not deform at any moment. When we look at simulations (b) and (c) with $k_{stiffness} = 0.1$ and $k_{stiffness} = 0.01$, respectively, we observe that the smaller the stiffness, the more the sphere deforms.

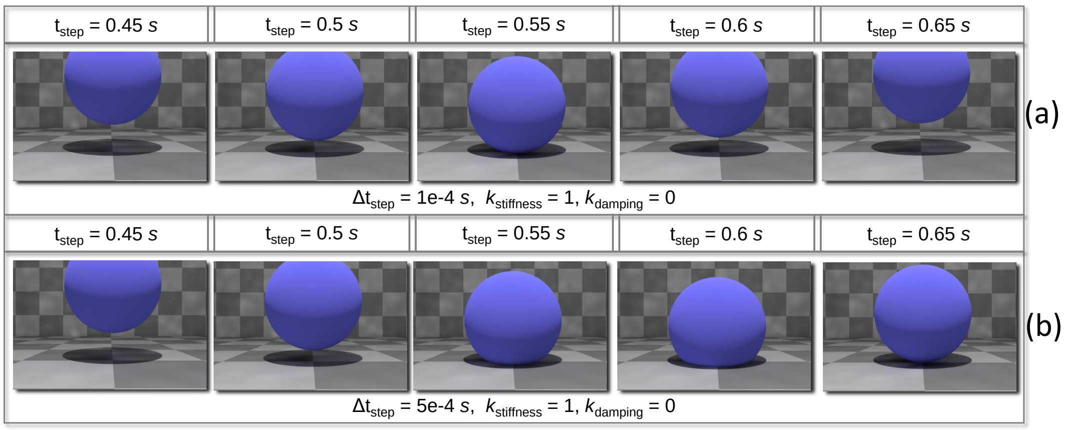
However, in Figure 6.8(b), maintaining $k_{stiffness} = 1$ and increasing the time step to 5e-4 s, the sphere begins deforming even with maximum stiffness constant. We conclude that the greater the time step, the less rigid the object becomes.

Another parameter influencing the body's rigidity is $k_{damping}$, in which a value of 1 gives a completely rigid body and damps all deformation without influencing global motion. We use this technique whenever we are not colliding



t_{step} = 0.45 s
t_{step} = 0.5 s
t_{step} = 0.55 s
t_{step} = 0.6 s
t_{step} = 0.65 s
(a)
Δt_{step} = 1e-4 s, $k_{stiffness}$ = 1, $k_{damping}$ = 0
(b)
Δt_{step} = 1e-4 s, $k_{stiffness}$ = 0.1, $k_{damping}$ = 0
(c)
Δt_{step} = 1e-4 s, $k_{stiffness}$ = 0.01, $k_{damping}$ = 0

Figure 6.7 – Three animations of a falling hollow sphere under gravity action. (a) The sphere collides and bounces in rigid form with $k_{stiffness} = 1$. (b) The sphere collides and bounces in deformed form with $k_{stiffness} = 0.1$. (c) The sphere collides and bounces in extreme deformed form with $k_{stiffness} = 0.01$.



t_{step} = 0.45 s
t_{step} = 0.5 s
t_{step} = 0.55 s
t_{step} = 0.6 s
t_{step} = 0.65 s
(a)
Δt_{step} = 1e-4 s, $k_{stiffness}$ = 1, $k_{damping}$ = 0
(b)
Δt_{step} = 5e-4 s, $k_{stiffness}$ = 1, $k_{damping}$ = 0

Figure 6.8 – Two animations of a falling hollow sphere under gravity action. (a) The sphere collides and bounces in rigid form with $k_{stiffness} = 1$ with a time step of 1e-4 s. (b) The sphere collides and bounces in deformed form with $k_{stiffness} = 1$ and increased time step of 5e-4 s.

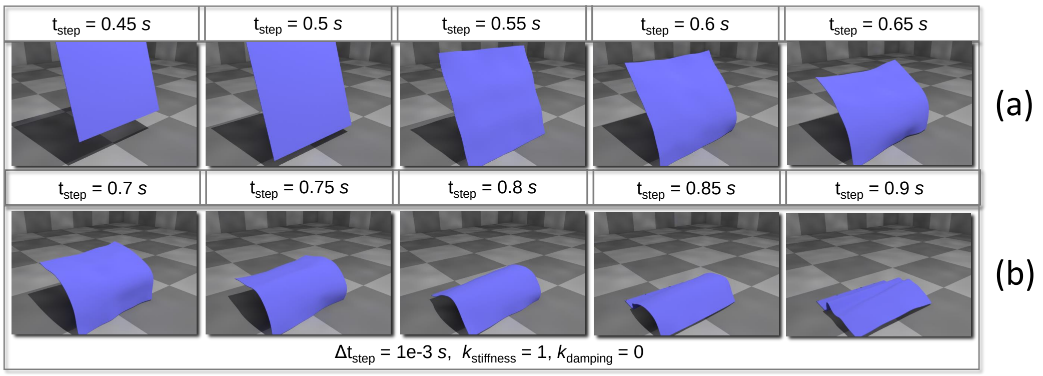


Figure 6.9 – Animation of a falling plate under gravity action. (a) The plate collides and (b) bounces in deformed form with $k_{damping} = 0$.

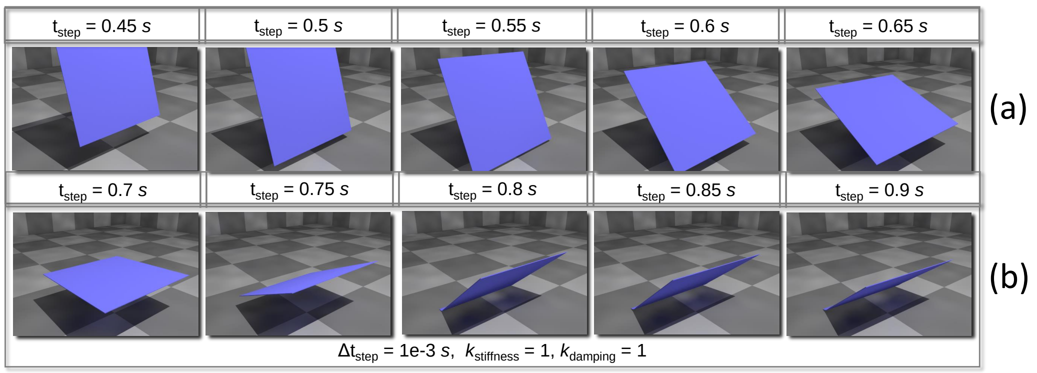


Figure 6.10 – Animation of a falling rigid body plate under gravity action. (a) The plate collides and (b) bounces in rigid form with $k_{damping} = 1$ and increased time step.

the object, inserting cohesive elements and duplicating nodes. Figure 6.9 shows a falling plate under gravity with $k_{damping} = 0$. At t = 0.55 s, the plate collides, but because no damping is applied, the plate completely deforms. When $k_{damping} = 1$ in Figure 6.10, the plate is treated as a rigid body and global motion survives.

The amount of cracks propagated in the mesh is influenced by the Normal Cohesive Strength parameter. Recall from Chapter 2 that “we determine if the principal stresses at each facet between two bulk elements exceed a limit for each of the nodes composing the element. Average stresses are computed to check for cohesive element insertion. We indicate that a facet is fractured if the stress exceeds a given threshold [8]”. In this case, we check if the principal stress at the node exceeds 90% of the maxmimum Normal Cohesive Strength parameter. The higher the Elastic Modulus of the object compared to the Normal Cohesive Strength, the more cracks will tend to propagate in the mesh discretization. Figure 6.11 demonstrates an example where we fixed and Elastic Modulus = 10e1 Pa. In the first row, the object breaks with a Normal Cohesive Strength smax = 5 mPa, while in the second row, the object breaks even more

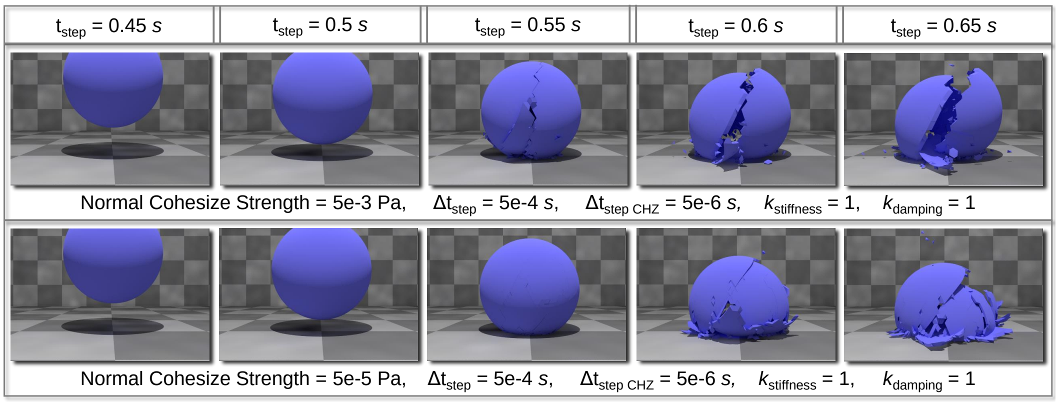

t_step = 0.45 s
t_step = 0.5 s
t_step = 0.55 s
t_step = 0.6 s
t_step = 0.65 s
Normal Cohesize Strength = 5e-3 Pa, Δt_step = 5e-4 s, Δt_step CHZ = 5e-6 s, k_stiffness = 1, k_damping = 1
Normal Cohesize Strength = 5e-5 Pa, Δt_step = 5e-4 s, Δt_step CHZ = 5e-6 s, k_stiffness = 1, k_damping = 1

Figure 6.11 – Two animations of a falling and breaking sphere under gravity action. (a) The sphere collides and breaks with smax = 5 mPa. (b) The sphere collides and breaks with smax = 500 mPa.

with an smax = 0.5 μPa.

Additional examples are shown in figures below. Figure 6.12 shows a brittle hollow sphere under gravity action and breaking when colliding with the floor. The mesh had a time step of 2 ms when in rigid body mode and 5 μs when in fracture mode. Initial material parameters are: initial velocity = 0 m/s, elastic modulus = 100 Pa, Poisson coefficient = 0.1, specific mass = 2400 kg/m3. Fracture energy materials are as follows: fracture energy G_I = 22 N/m, cohesive strength smax = 5 mPa, and shape parameter $\alpha = 2$. The sphere falls and breaks in half due to the level of coarsness of the mesh, time step size during insertion of cohesive elements (leading to smaller disaplacement of nodes when hitting the floor) and smaller proportion between elastic modulus and cohesive strength.

Figure 6.13 shows a brittle hollow bunny, composed of 69,668 nodes and 208,353 bulk elements, under gravity action and breaking when hit by a moving sphere object. The bunny mesh had a time step of 2 ms when in rigid body mode and 3 μs when in fracture mode. Initial material parameters are: initial velocity of the moving object = 10 m/s, elastic modulus = 6e-1 Pa, Poisson coefficient = 0.24, specific mass = 2400 kg/m3. Fracture energy materials are as follows: fracture energy G_I = 22 N/m, cohesive strength smax = 0.05 mPa, and shape parameter $\alpha = 2$. The bunny mesh fragments much due to the proportion of the elastic modulus with the cohesive strength.

Figure 6.14 shows a brittle glass plate under gravity action and breaking when hiting the ground. The plate mesh had a time step of 2 ms when in rigid body mode, and 5 μs when in fracture mode. Initial material parameters are: initial velocity of the moving object = 0 m/s, elastic modulus = 0.5 MPa, Poisson coefficient = 0.24, specific mass = 2400 kg/m3. Fracture energy

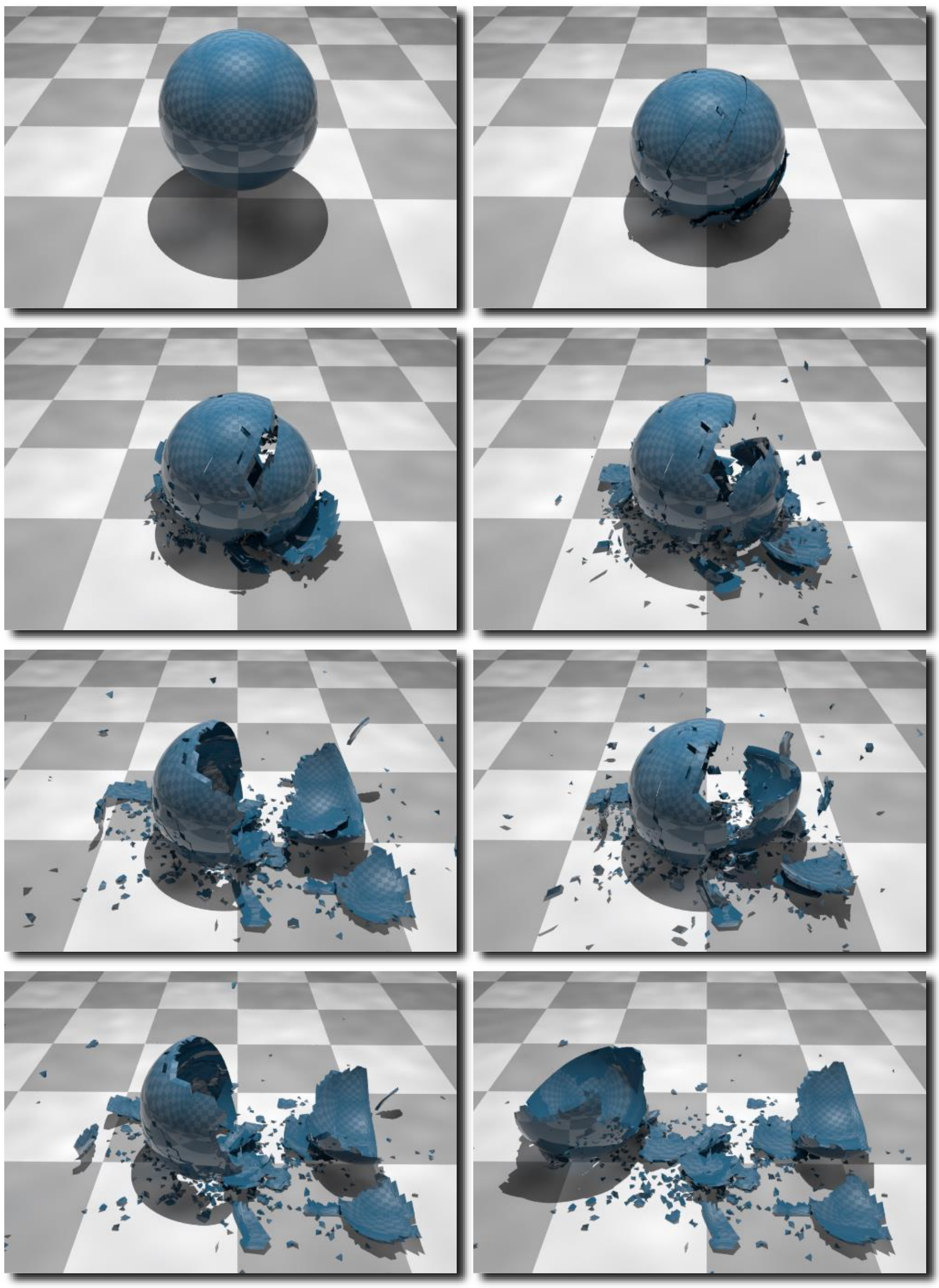


Figure 6.12 – Animation of a falling hollow sphere under gravity action and breaking when colliding with the floor.

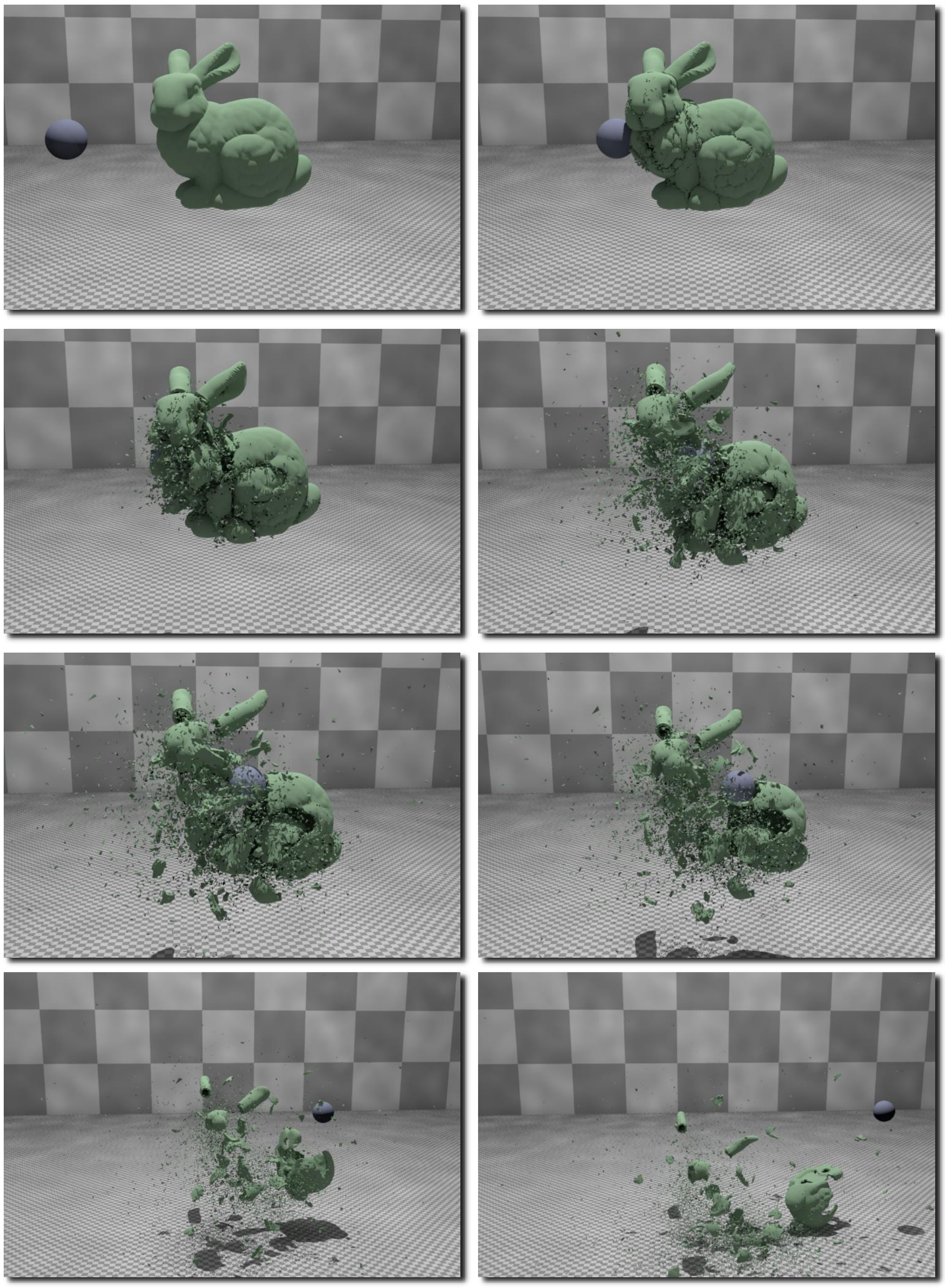


Figure 6.13 – Animation of a hollow bunny mesh hit by a sphere under velocity v = 10 m/s.

materials are as follows: fracture energy $G_I = 22e5$ N/m, cohesive strength smax = 50 Pa, and shape parameter $\alpha = 2$.

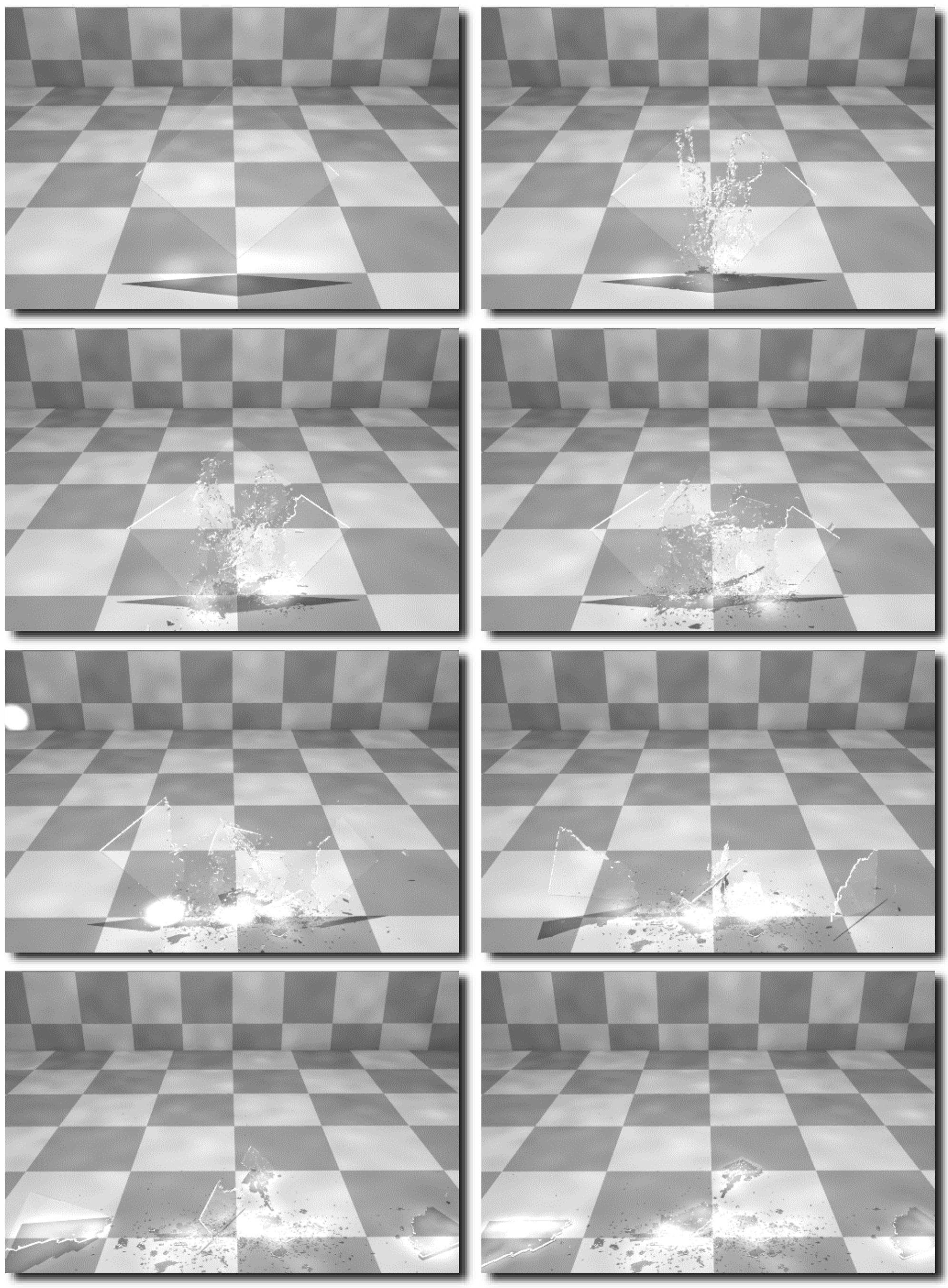


Figure 6.14 – Animation of a falling glass plate and breaking when colliding with the floor.